Techinical Description of the AutoRegister Components

Oliver Hinds <ohinds@orchardscientific.org>

Contents

1 Overview

This document explains the techinical details of the components produced for Phase One of the AutoRegister grant. The accompanying document `workflow.md` describes the installation and operation of the AutoRegister system.

2 Scout MRI pulse sequence

The AutoRegister Scout MRI pulse sequence is needed to acquire an image of a patient's head, which is appropriate for computing a spatial transformation between a previous or subsequent image of the same patient.

2.1 Repository

TODO

2.2 Details

The AutoRegister scout pulse sequence is based directly off of the gradient echo pulse sequence distributed by Siemens under the name `a_gre`. The scout has been successfully tested on Siemens baselines VB17A_???????? and VD13C_20121124. A scout for baseline VE11B_20150530 is under development.

For each platform version, the only change that needs to be made from the Siemens default pulse sequence is to change the ICE program filename to point to the AutoRegister MR Image reconstruction module. Specifically, clone the a_gre pulse sequence code to a new sequence `AutoRegisterScout` and change the line in a_gre.cpp: `rSeqExpo.setICEProgramFilename(...)` to point to the AutoRegister ICE Program, e.g. `%CustomerIceProgs%\\ohinds\\IceProgramAutoRegisterInterface`

2.3 Protocol

Early in the AutoRegister project, a series of test scans was conducted to determine a set of protocol parameters appropriate to act as a scout.

TODO describe the experiment and results

TODO list the protocol parameters

3 Registration module

The registration module is python software that runs on a computer external to the MRI system: currently a laptop in the scanner control room. The software receives an image from the MR Image reconstruction module, computes a spatial transformation, and sends the transformation back to the Image reconstruction module.

3.1 Repository

TODO

3.2 Environment

3.2.1 Python

To avoid version and package conflicts, the python software runs in a dedicated virtual environment produced by the `virtualenv` software. The workflow.md file describes the process of setting up the virtual environment, which is very simple.

The python libraries on which the Registration module depends are listed below.

- `numpy` for matrix math
- `nibabel` for reading and writing NIFTI images
- `nosetest` for running tests

3.2.2 `mri_robust_register`

Co-registration of scout images is accomplished using the tool `mri_robust_register` from the FreeSurfer software package. Instructions for installing FreeSurfer and configuring a shell environment suitable for running `mri_robust_register` are available at `http://freesurfer.net/`.

3.3 File formats

3.3.1 NIFTI

The NIFTI file format is widely used to store MR images, and `mri_robust_register` inherits NIFTI compatibility from FreeSurfer. The Registration module uses the `nibabel` python package to write out NIFTI files when MR images are received from the Image reconstruction module.

3.3.2 LTA

The LTA file format stores a linear transformation in text format. This is the format in which `mri_robust_register` stores computed transformations. The Registration module contains custom code to load a transformation from an LTA file.

3.4 Source

The source code for the Registration module is written in Python. It has been tested with Python version 2.7.

3.4.1 auto_register.py

The top-level file in the Registration module.

```
NAME
    auto_register - Main file and class for the autoregister application.

FILE
    /home/ohinds/projects/auto_register/src/auto_register.py

CLASSES
    __builtin__.object
        AutoRegister

    class AutoRegister(__builtin__.object)
     |  Methods defined here:
     |
     |  __init__(self, args)
     |      Initialize the autoregister application and helper modules.
     |
     |  check_for_input(self)
     |      Return the last character input, or None. If 'q' is seen, the
     |      autoregister application shuts down.
     |
     |  run(self)
     |      Main loop of the autoregister application.
     |
     |  shutdown(self)
     |      Shutdown the autoregister application. Stops the mainloop and
     |      tears down helper modules.

FUNCTIONS
    main(args)
        Main entry point
```

3.4.2 external_image.py

```
NAME
    external_image

FILE
```

```
    /home/ohinds/projects/auto_register/src/external_image.py

CLASSES
    __builtin__.object
        ExternalImage

    class ExternalImage(__builtin__.object)
     |  Methods defined here:
     |
     |  __init__(self, typename, format_def=[('magic', '5s'), ('headerVersion', 'i'), ('series
     |
     |  create_header(self, img, idx, nt, mosaic)
     |
     |  from_image(self, img, idx, nt, mosaic=True)
     |
     |  get_header_size(self)
     |
     |  get_image_size(self)
     |
     |  hdr_from_bytes(self, byte_str)
     |
     |  hdr_to_bytes(self, hdr_info)
     |
     |  make_img(self, in_bytes)
     |
     |  process_header(self, in_bytes)
     |
     |  process_image(self, in_bytes)
     |
     |  ----------------------------------------------------------------------
     |  Data and other attributes defined here:
     |
     |  struct_def = [('magic', '5s'), ('headerVersion', 'i'), ('seriesUID', '...

FUNCTIONS
    demosaic(mosaic, x, y, z)

    mosaic(data)

    sleep(...)
        sleep(seconds)

        Delay execution for a given number of seconds.  The argument may be
        a floating point number for subsecond precision.
```

3.4.3 `image_receiver.py`

3.4.4 `registered_image.py`

3.4.5 `tcpip_server.py`

3.4.6 `terminal_input.py`

3.4.7 `transform_sender.py`

3.5 Tools

3.5.1 vsend_nii

TODO

3.6 Tests

TODO

4 MR image reconstruction module

4.1 Repository

TODO

4.2 Details