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.

```
Help on module auto_register:

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.
     |
     |  ----------------------------------------------------------------------
     |  Data descriptors defined here:
     |
     |  __dict__
     |      dictionary for instance variables (if defined)
     |
     |  __weakref__
     |      list of weak references to the object (if defined)
```

```
FUNCTIONS
    main(args)
        Main entry point
```

3.4.2 external_image.py

```
Help on module external_image:

NAME
    external_image - Storage and I/O for an image received from an external sender.

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 descriptors defined here:
     |
     |  __dict__
     |      dictionary for instance variables (if defined)
     |
     |  __weakref__
```

```
 |      list of weak references to the object (if defined)
 |
 |  ----------------------------------------------------------------------
 |  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

```
Help on module image_receiver:

NAME
    image_receiver

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

DESCRIPTION
    Receive images sent to a TCP server. Also save the images and make
    the filename available through an external interface.

CLASSES
    __builtin__.object
        ImageReceiver

    class ImageReceiver(__builtin__.object)
     |  Run a TCP server, receive images, and save them.
     |
     |  Methods defined here:
     |
     |  __init__(self, args)
     |      Store arguments, determine first available image name so we don't
     |      overwrite existing images, and create a template image header for saving.
     |
     |  get_next_filename(self)
     |      If there is a new image file available, return it and remove the
     |      filename from those available.
```

```
 |
 |  is_running(self)
 |      Get whether the server is running.
 |
 |  process_data(self, sock)
 |      Callback to receive image data when it arrives.
 |
 |  save_nifti(self, img)
 |      Save a received image to a file.
 |
 |  start(self)
 |      Start the server to listen for incoming images.
 |
 |  stop(self)
 |      Stop listening for incoming images.
 |
 |  ----------------------------------------------------------------------
 |  Data descriptors defined here:
 |
 |  __dict__
 |      dictionary for instance variables (if defined)
 |
 |  __weakref__
 |      list of weak references to the object (if defined)

FUNCTIONS
    main(argv)
        Main entry. Just starts the server and waits for it to finish.

        USED IN STANDALONE MODE ONLY

    parse_args(args)
        Parse command line arguments.

        USED IN STANDALONE MODE ONLY

    sleep(...)
        sleep(seconds)

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

3.4.4 registered_image.py

```
Help on module registered_image:

NAME
    registered_image
```

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

CLASSES
    RegisteredImage

    class RegisteredImage
     |  Class that registers two images and stores info about the registration.
     |
     |  Methods defined here:
     |
     |  __init__(self, reference, movable, opts=None, verbose=False)
     |      Setup for performing registrations.
     |
     |  get_transform(self)
     |      Retrieve a 4x4 numpy matrix representing the most recently computed
     |      transform.
     |
     |  get_transform_filename(self)
     |      Return the name of the most recently computed transform.
     |
     |  register(self)
     |      Call out to the external program to register the reference and
     |      movable images.
     |
     |  ----------------------------------------------------------------------
     |  Class methods defined here:
     |
     |  check_environment(cls) from __builtin__.classobj
     |      Make sure that our environment is able to execute the registration
     |      program.
     |
     |  read_transform_file(cls, filename) from __builtin__.classobj
     |      Read and LTA file and parse the transformation it contains.

FUNCTIONS
    main(argv)
        During standalone operation, build all the arguments that would
        normally be passed in from the calling module and pass them into a
        RegisteredImage class instance.
```

3.4.5 tcpip_server.py

```
Help on module tcpip_server:

NAME
    tcpip_server - Threaded TCP/IP server that will notify a callback on a new connection.
```

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

CLASSES
    SocketServer.BaseRequestHandler
        ThreadedTCPRequestHandler
    SocketServer.TCPServer(SocketServer.BaseServer)
        ThreadedTCPServer(SocketServer.ThreadingMixIn, SocketServer.TCPServer)
    SocketServer.ThreadingMixIn
        ThreadedTCPServer(SocketServer.ThreadingMixIn, SocketServer.TCPServer)

    class ThreadedTCPRequestHandler(SocketServer.BaseRequestHandler)
     |  Methods defined here:
     |
     |  __init__(self, callback, *args, **keys)
     |
     |  handle(self)
     |
     |  ----------------------------------------------------------------------
     |  Methods inherited from SocketServer.BaseRequestHandler:
     |
     |  finish(self)
     |
     |  setup(self)

    class ThreadedTCPServer(SocketServer.ThreadingMixIn, SocketServer.TCPServer)
     |  Method resolution order:
     |      ThreadedTCPServer
     |      SocketServer.ThreadingMixIn
     |      SocketServer.TCPServer
     |      SocketServer.BaseServer
     |
     |  Methods defined here:
     |
     |  __init__(self, address, callback)
     |
     |  is_running(self)
     |
     |  ----------------------------------------------------------------------
     |  Methods inherited from SocketServer.ThreadingMixIn:
     |
     |  process_request(self, request, client_address)
     |      Start a new thread to process the request.
     |
     |  process_request_thread(self, request, client_address)
     |      Same as in BaseServer but as a thread.
     |
     |      In addition, exception handling is done here.
     |
```

```
|  ----------------------------------------------------------------------
|  Data and other attributes inherited from SocketServer.ThreadingMixIn:
|
|  daemon_threads = False
|
|  ----------------------------------------------------------------------
|  Methods inherited from SocketServer.TCPServer:
|
|  close_request(self, request)
|      Called to clean up an individual request.
|
|  fileno(self)
|      Return socket file number.
|
|      Interface required by select().
|
|  get_request(self)
|      Get the request and client address from the socket.
|
|      May be overridden.
|
|  server_activate(self)
|      Called by constructor to activate the server.
|
|      May be overridden.
|
|  server_bind(self)
|      Called by constructor to bind the socket.
|
|      May be overridden.
|
|  server_close(self)
|      Called to clean-up the server.
|
|      May be overridden.
|
|  shutdown_request(self, request)
|      Called to shutdown and close an individual request.
|
|  ----------------------------------------------------------------------
|  Data and other attributes inherited from SocketServer.TCPServer:
|
|  address_family = 2
|
|  allow_reuse_address = True
|
|  request_queue_size = 5
|
|  socket_type = 1
|
```

```
 |  ----------------------------------------------------------------------
 |  Methods inherited from SocketServer.BaseServer:
 |
 |  finish_request(self, request, client_address)
 |      Finish one request by instantiating RequestHandlerClass.
 |
 |  handle_error(self, request, client_address)
 |      Handle an error gracefully.  May be overridden.
 |
 |      The default is to print a traceback and continue.
 |
 |  handle_request(self)
 |      Handle one request, possibly blocking.
 |
 |      Respects self.timeout.
 |
 |  handle_timeout(self)
 |      Called if no new request arrives within self.timeout.
 |
 |      Overridden by ForkingMixIn.
 |
 |  serve_forever(self, poll_interval=0.5)
 |      Handle one request at a time until shutdown.
 |
 |      Polls for shutdown every poll_interval seconds. Ignores
 |      self.timeout. If you need to do periodic tasks, do them in
 |      another thread.
 |
 |  shutdown(self)
 |      Stops the serve_forever loop.
 |
 |      Blocks until the loop has finished. This must be called while
 |      serve_forever() is running in another thread, or it will
 |      deadlock.
 |
 |  verify_request(self, request, client_address)
 |      Verify the request.  May be overridden.
 |
 |      Return True if we should proceed with this request.
 |
 |  ----------------------------------------------------------------------
 |  Data and other attributes inherited from SocketServer.BaseServer:
 |
 |  timeout = None

FUNCTIONS
    handler_factory(callback)

    process_data_callback(callback, sock)
```

3.4.6 terminal_input.py

```
Help on module terminal_input:

NAME
    terminal_input - class to interact with the user through a terminal.

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

CLASSES
    __builtin__.object
        TerminalInput

    class TerminalInput(__builtin__.object)
     |  Methods defined here:
     |
     |  __init__(self, disabled)
     |
     |  get_char(self)
     |
     |  run(self)
     |
     |  start(self)
     |
     |  stop(self)
     |
     |  ----------------------------------------------------------------------
     |  Data descriptors defined here:
     |
     |  __dict__
     |      dictionary for instance variables (if defined)
     |
     |  __weakref__
     |      list of weak references to the object (if defined)
```

3.4.7 transform_sender.py

```
Help on module transform_sender:

NAME
    transform_sender - class to transmit an affine transform over a socket.

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

CLASSES
    __builtin__.object
```

```
    TransformSender

class TransformSender(__builtin__.object)
 |  Methods defined here:
 |
 |  __init__(self, host, port)
 |
 |  clear_state(self)
 |
 |  process_data(self, sock)
 |
 |  send(self, transform)
 |
 |  set_state(self, state)
 |
 |  start(self)
 |
 |  stop(self)
 |
 |  ----------------------------------------------------------------------
 |  Data descriptors defined here:
 |
 |  __dict__
 |      dictionary for instance variables (if defined)
 |
 |  __weakref__
 |      list of weak references to the object (if defined)
```

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