An ITK-Based Framework for 2D-3D Registration with Multiple Fixed Images

Release 1.00

Álvaro Bertelsen[1] and Diego Borro[1]

May 23, 2011

[1]CEIT and Tecnun, University of Navarra, 20018, San Sebastián, Spain

Abstract

This document describes an extension of the Insight Toolkit (ITK, www.itk.org) for 2D-3D registration with multiple fixed images, that is, registration of a three-dimensional dataset to a group of fixed planar projections. 2D-3D registration is possible with ITK's standard classes but with several limitations: the number of fixed images is restricted to one and the moving image's orientation axes are ignored, which greatly complicates the registration definition. Both problems are solved by the proposed framework, which permits multi-resolution intensity-based registration with an arbitrary number of fixed images, with all images defined in any orientation. In addition, the framework provides implementations of the Normalized Gradient Correlation and Pattern Intensity metrics, which are commonly used in 2D-3D registration but were not present in ITK. This article gives a detailed description of the proposed framework, along with examples that show its capabilities in registrations of real and simulated images of the spine.

Contents

1 Introduction

Registration techniques of pre-operative and intra-operative images are prerequisites for a large variety of image guided interventions, such as radiotherapy planning, minimally invasive surgery and endoscopic procedures. Among these techniques, 2D-3D registration consists of the matching of a moving 3D dataset to one or more fixed 2D projections. Most of the time, the 3D data comes in the form of a CT or MR scan, while 2D images are x-rays acquired with a C-arm. A recent study by Markelj et al. [2] gives a comprehensive review of the large amount of strategies developed by different research groups to solve this problem. Roughly, 2D-3D registration algorithms can be grouped according to the type of data used –features, intensity or gradients– and the dimensional correspondence used to evaluate the images' similarity –projection, back-projection and reconstruction– where projective intensity-based registration is the most common strategy. Algorithms of this type generate Digitally Reconstructed Radiographs (DRR) projecting the 3D image onto the x-rays' planes. Similarities between the DRRs and x-rays are then evaluated using a metric which compares their intensities voxel-by-voxel. The 3D image is transformed and the DRRs are regenerated until the similarity metric reaches its maximum. As some spatial information is inherently lost on the DRR generation, multiple projections acquired in different orientations are used to ensure that all directions can be observed. Traditionally, two projections are used, which are acquired in the Antero-Posterior (AP) and Lateral (LAT) orientations.

The Insight Toolkit (ITK, `www.itk.org`), a widely used open-source software library for medical image processing, gives limited support for 2D-3D registration. Using the standard ITK classes, it is possible to implement projective intensity-based registrations, but with considerable limitations:

- Only one fixed image can be included in the registration.
- The orientation axes of the moving image are ignored, which complicates the geometric definition of the registration.
- Only one ray-casting algorithm is available for DRR generation, which only supports rigid transformations.
- No support for multi-threading is available.

This work proposes a set of classes based on ITK 3.20.0 which solves the first two problems listed above. The proposed framework allows implementation of registrations which accept one, two or more fixed images and register the moving dataset to all of the fixed images simultaneously, considering the orientation information present in the moving image. Although the proposed framework does not address the latter two problems –non-rigid transformations and multi-threading support– the interested reader should consult the work by Steininger et al. [5], who proposed an ITK extension which solved the aforementioned issues (`http://ibia.umit.at/ResearchGroup/Phil/web/Simple2D3DRegistrationFramework.html`).

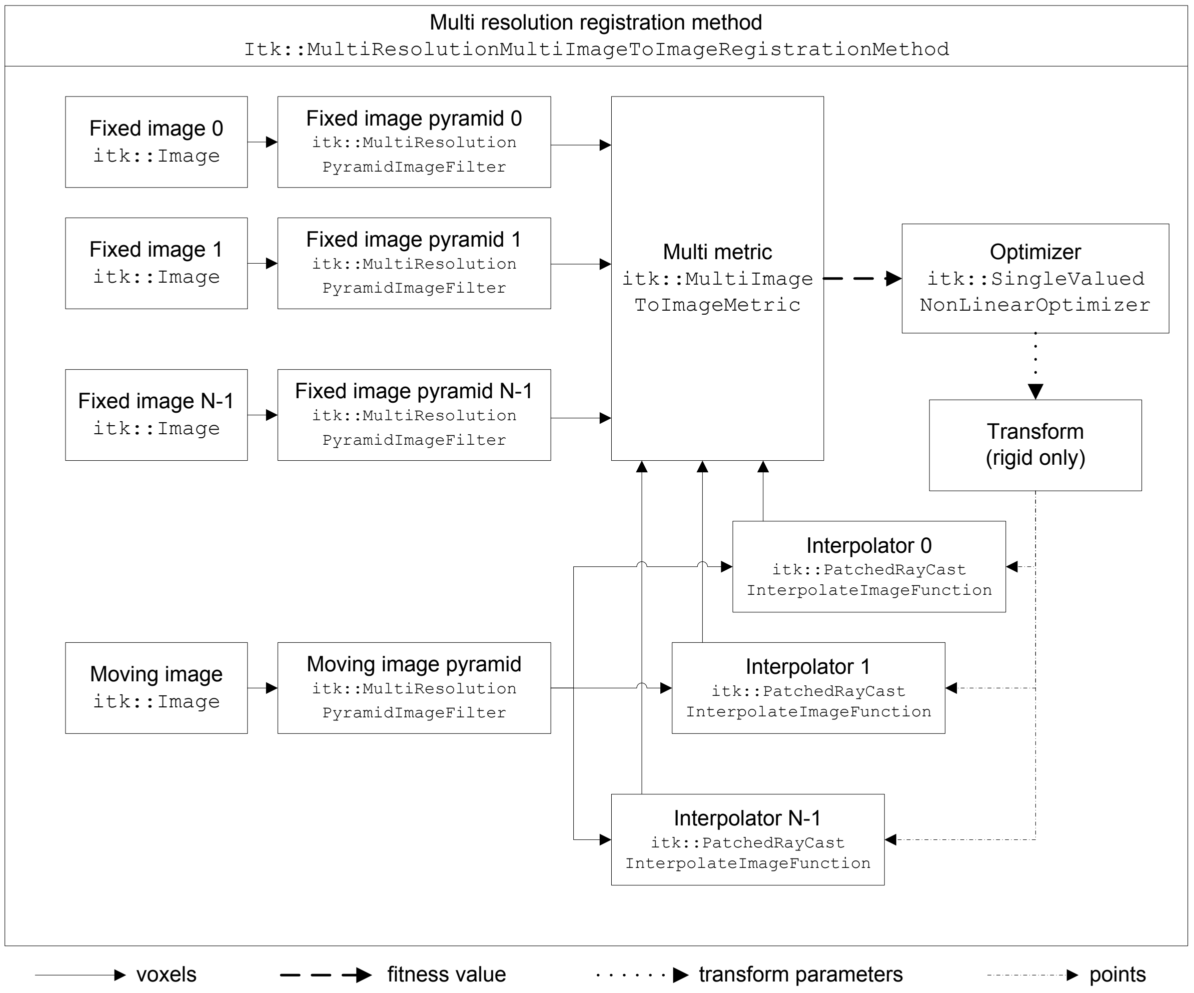

Multi resolution registration method
Itk::MultiResolutionMultiImageToImageRegistrationMethod
Fixed image 0
itk::Image
Fixed image pyramid 0
itk::MultiResolution PyramidImageFilter
Fixed image 1
itk::Image
Fixed image pyramid 1
itk::MultiResolution PyramidImageFilter
Fixed image N-1
itk::Image
Fixed image pyramid N-1
itk::MultiResolution PyramidImageFilter
Multi metric
itk::MultiImage ToImageMetric
Optimizer
itk::SingleValued NonLinearOptimizer
Transform (rigid only)
Interpolator 0
itk::PatchedRayCast InterpolateImageFunction
Moving image
itk::Image
Moving image pyramid
itk::MultiResolution PyramidImageFilter
Interpolator 1
itk::PatchedRayCast InterpolateImageFunction
Interpolator N-1
itk::PatchedRayCast InterpolateImageFunction
voxels
fitness value
transform parameters
points

Figure 1: Connection diagram of the proposed framework's classes. The main differences between this framework and the ITK's current one are the use of the `itk::MultiImageToImageMetric` class instead of the standard metric object and the use of multiple interpolators. Also note that registration is encapsulated in the new `MultiResolutionMultiImageToImageRegistrationMethod`.

This article is organised as follows: Section 2 gives a brief theoretical definition of the registration problem followed by a detailed description of the proposed framework's classes and interface. Section 3 describes a number of example applications that demonstrate the framework's capabilities in registration of real and simulated data. Finally, conclusions and future work guidelines are presented on Section 4.

2 Framework for intensity-based registration

The 2D-3D intensity-based registration problem is defined as follows. Let M be the moving 3D image and F_k with $k = 0..N-1$ the set of N fixed 2D images. Following ITK's convention, let $\mathcal{T}$ be a geometrical transform which takes a point $\mathbf{x}$ from the fixed image's physical coordinate space to the moving image's physical coordinate space. For each fixed image F_k a corresponding 2D image $\mathcal{P}_k^{\mathcal{T}M}$ exists, which is the projection of the moving image onto the plane of F_k after application of transform $\mathcal{T}$. In other words, $\mathcal{P}_k^{\mathcal{T}M}$ is the k-th DRR, mathematically defined as

$$\mathcal{P}_k^{\mathcal{T}M}(\mathbf{x}) = \int_0^1 M(\mathcal{T}(\mathbf{e}_k + \lambda(\mathbf{x} - \mathbf{e}_k)))\mathrm{d}\lambda \tag{1}$$

that is, the intensity value at point $\mathbf{x}$ of $\mathcal{P}_k^{\mathcal{T}M}$ equal the integral of the moving image M intensities sampled along the ray that goes from $\mathbf{x}$ to the projection's focal point e_k, after application of transform $\mathcal{T}$. The ray equation is parametrised using the variable λ: if λ is 0 the ray is in $\mathbf{x}$, if λ is 1 the ray is the focal point e_k. It is assumed that a direct integration of the voxel values of M gives a DRR useful for registration purposes, so no intensity corrections are needed.

Let S_k be a metric which computes the similarity between F_k and $\mathcal{P}_k^{\mathcal{T}M}$ on a defined set of points $\mathbf{x}_{Fk}$. The problem of intensity-based 2D-3D registration consists of finding the optimal transform $\dot{\mathcal{T}}$ which maximises the sum of all similarity metrics S_k

$$\begin{aligned}\dot{\mathcal{T}} &= \arg\max_{\mathcal{T}} \sum_{k=1}^{K} S_k \\ &= \arg\max_{\mathcal{T}} \sum_{k=1}^{K} S_k(F_k, \mathcal{P}_k^{\mathcal{T}M}, \mathbf{x}_{Fk})\end{aligned} \tag{2}$$

In terms of programming, ITK provides classes for all objects involved in the registration: transform, metric, optimizer and interpolator, used to sample the moving image's values on non-grid positions. All of these objects are plugged into a registration method class, which makes all the necessary internal connections and coordinates the optimization to find the optimal transform parameters. Usually, the registration method object is an instance of `itk::ImageRegistrationMethod`, which expects a generic transform, interpolator and a metric object subclassed from `itk::SingleValuedCostFunction`, as it must return a single real value as a quality measure of the match between the fixed and moving images. For the same reason, the used optimizer must be designed to find the optimum of single-valued cost function and must be subclassed from `itk::SingleValueNonLinearOptimizer`. Multi-resolution registration is possible by using the `itk::MultiResolutionImageRegistrationMethod`, which requires filters of class `itk::MultiResolutionPyramidImageFilter` to generate the downsampled versions of the moving and fixed images. For 2D-3D registration, ITK provides the `itk::RayCastInterpolateImageFunction` for generation of DRR using the standard ray-casting algorithm.

Any developer familiar with ITK's paradigm should be able to adjust to the proposed one with minimal effort, as the former has been largely respected. Registration objects work in almost identical way, although they are instances of the new `itk::MultiImageToImageRegistrationMethod` and `MultiResolutionMultiImageToImageRegistrationMethod` classes. Its main difference with the standard classes is the use of multiple fixed images, pyramid filters and interpolators instead of the single ones needed in ITK. Optimizers remain as subclasses of `SingleValuedNonLinearOpitmizer`, but metric objects have been replaced by the new `MultiImageToImageMetric` class which, like in standard ITK, return a single real value, but accept multiple fixed images as input instead of the single one permitted by ITK's metrics. Concrete implementations of the Gradient Difference, Normalized Gradient Correlation and Pattern Intensity metrics for multiple fixed images are provided, which are widely-used in 2D-3D registration. In addition, a new ray-casting interpolator class named `itk::PatchedRayCastInterpolateImageFunction` is provided, which fixes some of the bugs present in the original implementation. A schematic of the framework's classes is shown in Figure 1, which shows how objects are connected for a typical registration problem.

Detailed descriptions of all classes present in the proposed framework are given in the sections below.

2.1 Registration methods

The main class of the framework is the itk::MultiImageToImageRegistrationMethod, which coordinates all objects required in the registration process, which are the images, transform, metric, optimizer and interpolators. As itk::MultiImageToImageRegistrationMethod was subclassed from itk::ProcessObject, a call to the Update method is all that is needed to execute the registration once its components have been plugged in. Multi-resolution registration can be done by using the itk::MultiResolutionMultiImageToImageRegistrationMethod, a subclass of itk::MultiImageToImageRegistrationMethod.

The moving image, transform and optimizer can be set on a MultiImageToImageRegistrationMethod object using the standard SetMovingImage, SetTransform and SetOptimizer methods, also present in ITK's original registration classes. Addition of the metric requires creation of an itk::MultiImageToImageMetric instance, which should be later plugged into the registration object using the SetMultiMetric method. Adding the fixed images, regions, masks and interpolators is different, as multiple objects need to be plugged in rather than just one. The proposed framework offers two different ways to do this. The first and recommended way is to create each object individually and plug them into the registration using the corresponding Add method. The itk::MultiImageToImageRegistrationMethod has implementations of the AddFixedImage, AddFixedImageRegion, AddFixedImageMask and AddInterpolator methods for insertion of fixed images, regions, masks and interpolators respectively. For example, the code to add two fixed images should be as follows

```
typedef itk::Image<short,3>  FixedImageType;
typedef itk::Image<short,3>  MovingImageType;

typedef itk::MultiImageToImageRegistrationMethod<FixedImageType,
                                                 MovingImageType> RegistrationType;

const unsigned int FImgTotal = 2;

RegistrationType::Pointer registration = RegistrationType::New();

for( unsigned int f=0; f<FImgTotal; f++ )
  {
  FixedImageType::Pointer fixedImage;

// do something here to read or create a fixed image

  registration->AddFixedImage( fixedImage );
  }
```

The second way to define fixed images, regions and interpolators is to store them in std::vector objects and plug them into the registration using SetFixedMultiImage, SetFixedMultiImageRegion, SetFixedMultiImageMask or SetMultiInterpolator methods respectively. The code to add a set of fixed images should look like the written below:

```
RegistrationType::FixedMultiImageType fixedMultiImage;
```

```
for( unsigned int f=0; f<FImgTotal; f++ )
  {
  FixedImageType::ConstPointer fixedImage;

// do something here to read or create a fixed image

  fixedMultiImage.push_back( fixedImage );
  }

registration->SetFixedMultiImage( fixedMultiImage );
```

Usually, the elements of these types of arrays are `const` so care must be taken to store them correctly. The `MultiInterpolatorType` has an additional problem: its elements are of type `itk::InterpolateImageFunction::Pointer` so, if this type of array was created, the user should define the interpolator, cast it to `itk::InterpolateImageFunction::Pointer`, store it and finally plug it into the registration method. All these steps can be avoided if the `AddInterpolator` method is used instead, which can receive a `Pointer` of any subclass of `itk::InterpolateImageFunction` and, normally, leads to simpler code. As an example, compare the code snippet given below, which uses the `AddInterpolator` method,

```
typedef itk::PatchedRayCastInterpolateImageFunction<MovingImageType,
                                                    double> InterpolatorType;

for( unsigned int i=0; i<FImgTotal; f++ )
  {
  InterpolatorType::Pointer interpolator = InterpolatorType::New();

// here, code to define the interpolator's focal point, transform, etc.

  registration->AddInterpolator( interpolator );
  }
```

with the following one, which uses the `SetMultiInterpolator` method.

```
typedef itk::PatchedRayCastInterpolateImageFunction<MovingImageType,
                                                    double> InterpolatorType;

typedef RegistrationType::BaseInterpolatorPointer  BaseInterpolatorPointer;
typedef RegistrationType::MultiInterpolatorType    MultiInterpolatorType;

MultiInterpolatorType multiInterpolator;

for( unsigned int i=0; i<FImgTotal; f++ )
  {
  InterpolatorType::Pointer interpolator = InterpolatorType::New();

// here, code to define the interpolator's focal point, transform, etc.

  multiInterpolator.push_back( static_cast<BaseInterpolatorPointer>( interpolator ) );
  }
registration->SetMultiInterpolator( multiInterpolator );
```

Although both snippets implement the same function –addition of multiple interpolators to the registration method– the first one is much simpler.

The `itk::MultiImageToImageRegistrationMethod` class and its subclasses expect that the number of defined regions, masks and interpolators is exactly equal to the number of fixed images. However, if no regions have been set for the fixed images, `itk::MultiImageToImageRegistrationMethod` will use their buffered regions by default. Also, if no masks are defined, they will be ignored during registration. It is also possible to define masks for only some of the fixed images, by calling the `AddFixedImageMask` method with a `NULL` argument for the ones that should not be masked.

The `itk::MultiResolutionMultiImageToImageRegistrationMethod` implements multi-resolution registration with multiple fixed images. The class interface was kept as similar as possible to the available `itk::MultiResolutionImageRegistrationMethod`, so the user must provide the filters that generate the images for each resolution level. The filter for the moving image is set with the `SetMovingImagePyramid` method and the ones used for the fixed images are set with the `AddFixedImagePyramid` or `SetFixedMultiImagePyramid` methods. The number of resolution levels can be set using the `SetNumberOfLevels` method, which will define a default schedule for the different resolution levels. Finer control over the amount of blurring and downsampling is achieved with the `SetSchedules` method, which receives two matrices –one for the moving image and one for the fixed ones– whose elements define the downsampling factor applied in each dimension (column) for each resolution level (row). In 2D-3D registration, using `SetSchedules` is always recommended, as the fixed images' downsampling factor along the slice direction must be equal to 1 at all resolution levels (in other words, all elements on the third column of the fixed images' schedule matrix must be equal to 1).

Finally, note in the code snippets given above that both the fixed and moving images have been defined with the same number of dimensions, three. Despite that the fixed images are usually 2D, they must be defined as 3D images with a single slice, to ensure that ITK handles their orientation and origin information correctly.

2.2 Transform

When the `itk::PatchedRayCastInterpolateImageFunction` class is used for 2D-3D registration, the framework can only use rigid transforms such as `itk::TranslationTransform` and `itk::Euler3DTransform`. Despite this limitation, the proposed classes can still be used on plenty of applications, as 2D-3D registration is commonly used on rigid body structures such as the vertebrae, skull, pelvis and femur [2].

2.3 Metric

Equation 2 states that the degree of similarity in registrations with multiple fixed images can be calculated as the sum of individual similarity metrics between the moving image and each of the fixed ones. To implement this functionality, a new class named `itk::MultiImageToImageMetric` was written, which internally stores an array of individual metrics and returns their sum when its value is requested by the `GetValue` method.

Like the existing `itk::ImageToImageMetric` class, `itk::MultiImageToImageMetric` was derived from `itk::SingleValuedCostFunction`, so it could be easily incorporated into the standard ITK's registration framework. Hence, the standard `GetValue`, `GetDerivatives` and `GetValueAndDerivatives` methods are present. Calls to `GetValue` return the sum of the internal array of metrics while `GetDerivatives` return an estimation of the functions' derivatives using the central differences method, with a variable step size set with the `SetDerivativeDelta` method. `GetDerivatives` is defined as `virtual`, so developers can replace it with a more efficient method, as the central differences method requires multiple calls to `GetValue`, which can be slow and inaccurate for some applications.

The `itk::MultiImageToImageMetric` and its internal metric type `itk::ImageToImageMetric` are defined as purely virtual and cannot be instantiated. Concrete implementations of similarity metrics are implemented by the `itk::MultiImageToImageMetric` subclasses, which are responsible for redefinition of the internal metric type. Currently, the proposed framework has implementations of the following metrics:

- `itk::MeanSquaresMultiImageToImageMetric`
- `itk::GradientDifferenceMultiImageToImageMetric`
- `itk::NormalizedGradientCorrelationMultiImageToImageMetric`
- `itk::PatternIntensityMultiImageToImageMetric`

Mean Squares

The `itk::MeanSquaresMultiImageToImageMetric` offers an implementation of the well-known average of squared differences metric, based on the sum of multiple instances of `itk::MeanSquaresImageToImageMetric`. This metric requires both images to have similar intensity scales to work properly, which is very difficult to achieve in 2D-3D registration using DRRs. Thus, this function serves more as an example of how `itk::MultiImageToImageMetric` must be subclassed rather than for actual use.

Although `itk::MeanSquaresMultiImageToImageMetric` has an analytical formula for its gradient, it is not used when `GetDerivative` is called. The `itk::MeanSquaresMultiImageToImageMetric` gradient calculation assumes that valid points are only found in the overlapping regions between the fixed and moving images which, in the case of 2D-3D registration, are non-existent. Thus, the available analytical formula could not be used and was replaced by the finite differences method, which, despite being less efficient, makes no assumptions about overlap and calculates a correct estimate of the function's gradient.

Gradient Difference

The `itk::GradientDifferenceMultiImageToImageMetric` implements the Gradient Difference metric proposed by Penney et al. [4] for multiple fixed images. The proposed framework also includes a version for single fixed image registration named `itk::GradientDifferenceSingleImageToImageMetric`. Gradient Difference estimates the degree of similarity based on the image obtained by subtraction between the gradients of the fixed and projected images. Mathematically, it is defined as

$$S_k = \sum_{\mathbf{x}_{Fk}} \frac{\sigma_{ik}^2}{\sigma_{ik}^2 + GD_k^i(\mathbf{x}_{Fk})} + \sum_{\mathbf{x}_{Fk}} \frac{\sigma_{jk}^2}{\sigma_{jk}^2 + GD_k^j(\mathbf{x}_{Fk})} \tag{3}$$

where GD_k^i and GD_k^j are the differences between the gradients of fixed image k and the ones of the moving image's projection $\mathcal{P}_k^{\mathcal{T}M}$ along i and j, defined as the first and second directions in the fixed image's coordinate system.

$$GD_k^i = \frac{\delta F_k}{\delta i} - \frac{\delta \mathcal{P}_k^{\mathcal{T}M}}{\delta i} \tag{4}$$

$$GD_k^j = \frac{\delta F_k}{\delta j} - \frac{\delta \mathcal{P}_k^{\mathcal{T}M}}{\delta j} \tag{5}$$

Terms σ_{ik}^2 and σ_{jk}^2 are the variances from the gradient images of F_k

$$\sigma_{ik}^2 = var(\frac{\delta F_k}{\delta i}) \tag{6}$$

$$\sigma_{jk}^2 = var(\frac{\delta F_k}{\delta j}) \tag{7}$$

Gradient Difference reduces the effects of low frequency intensity changes, such as the ones introduced by soft tissue, by using the images' gradients rather than their intensities. In addition, its reciprocal form (differences are placed in the function's denominator) makes it strong against thin artefacts with strong intensities, such as surgical instruments present in the images [4].

Note that the framework includes an implementation of the Gradient Difference metric despite that ITK already has one. The main reason is that the original implementation, called `itk::GradientDifferenceImageToImageMetric` always pre-computes the moving image gradient. If it was used in a metric for multiple fixed images, ITK's implementation would compute multiple identical versions of this gradient and waste considerable amounts of memory by storing them.

Normalized Gradient Correlation

`itk::NormalizedGradientCorrelationMultiImageToImageMetric` implements the Normalized Gradient Correlation metric for multiple fixed images, based on the `itk::NormalizedGradientCorrelationImageToImageMetric` class also included in the proposed framework. This metric computes the average of the normalized cross-correlation between the gradients of the fixed and projected images computed along directions i and j [1].

Normalized Gradient Correlation is defined as

$$S_k = NCC\left(\frac{\partial F_k}{\partial i}, \frac{\mathcal{P}_k^{\mathcal{T}M}}{\partial i}, \mathbf{x}_{Fk}\right) + NCC\left(\frac{\partial F_k}{\partial j}, \frac{\mathcal{P}_k^{\mathcal{T}M}}{\partial j}, \mathbf{x}_{Fk}\right) \tag{8}$$

with $NCC(F, M, \mathbf{x})$ the Normalized Cross Correlation between images F and M evaluated on points $\mathbf{x}$

$$NCC(F, M, \mathbf{x}) = \frac{\sum_{\mathbf{x}} F(\mathbf{x})M(\mathbf{x})}{\sqrt{\sum_{\mathbf{x}} F(\mathbf{x})^2}\sqrt{\sum_{\mathbf{x}} M(\mathbf{x})^2}} \tag{9}$$

Normalized Gradient Correlation, like Gradient Difference, is insensitive to low-frequency variations introduced by soft tissue, as images' gradients are used to compute the metric rather than their intensities.

Pattern Intensity

The `itk::PatternIntensityMultiImageToImageMetric` implements the Pattern Intensity metric proposed by Weese et al. [10] for multiple fixed images. Like the two metrics introduced in the previous sections, its version for registrations with a single fixed image is also included in the proposed framework under the name `itk::PatternIntensityImageToImageMetric`. Pattern intensity characterises the structures found on the difference image ID_k, obtained after subtraction of the fixed image F_k and the projection $\mathcal{P}_k^{\mathcal{T}M}$. If the moving image is not registered properly, areas with large intensity variations, that is 'structures', will appear in the difference image ID_k. In turn, structures will vanish if registration is correct, leaving areas with low changes in intensity. The pattern intensity metric assigns values to the amount of structures found on a small moving kernel, according to the following formula

$$S_k = \sum_{\mathbf{x}_{Fk}} \sum_{\mathbf{u}_{Fk}} \frac{\sigma^2}{\sigma^2 + (ID_k(\mathbf{x}_{Fk}) - ID_k(\mathbf{u}_{Fk}))^2} \tag{10}$$

$$|\mathbf{x}_{Fk} - \mathbf{u}_{Fk}|^2 < r^2 \tag{11}$$

In Equation 10, term $\mathbf{x}_{Fk}$ represents the coordinates of each voxel taken into account for the metric calculation and $\mathbf{u}_{Fk}$, the coordinates of a moving kernel of radius r centered on voxel $\mathbf{x}_{Fk}$. This kernel samples the image ID_k, obtained from the subtraction of the fixed and projected images

$$ID_k = F_k - \mathcal{P}_k^{\mathcal{T}M} \tag{12}$$

Also, terms r and σ^2 are user-defined parameters. Parameter r defines the radius of the moving kernel. Parameter σ^2 defines the sensitivity which determines if a intensity variation should be considered a structure or not. These parameters can be set using the `SetRadius` and `SetSigma` methods. The default value is 3 voxels for r and 10 for σ.

The Pattern Intensity metric, differs from Gradient Difference and Normalized Gradient Correlation in that it does not compute the images' gradients. Instead, it uses direct subtraction between intensities of the fixed and projected images, but its moving kernel restricts the metric to the local features rather than the global ones, preventing soft tissue intensity variations from affecting the metric values. Like Gradient Difference, it also has a reciprocal form, which makes it strong against thin outliers like surgical instruments.

2.4 Optimizer

The proposed framework works with any optimizer derived from the `itk::SingleValuedNonLinearOptimizer` class. Van der Bom et al. [9] experimented with different combinations of metrics and optimizers for registration of a CT scan to a single x-ray of a human skull, demonstrating that proper combinations can considerably affect the registration performance. The best performing combination was the Normalized Gradient Correlation metric paired with a conjugate gradient descent optimization algorithm, such as the one implemented by the `itk::FRPROptimizer` class. However, it should be noted that registration problems are highly specific and the best combination of metric and optimizer for one may not be the best for another.

2.5 Interpolator

The proposed framework includes a new interpolator class, named `itk::PatchedRayCastInterpolateImageFunction`, which projects a 3D image onto a plane using the ray casting algorithm, integrating the intensity values along rays that go from all voxels on the imaging plane to a defined focal point, as defined in Equation 1. Ray casting offers a simplified model of the x-ray formation process, ignoring effects such as scattering, and has become a widely used algorithm for DRR generation.

The `itk::PatchedRayCastInterpolateImageFunction` is based on the existing `itk::RayCastInterpolateImageFunction`, which has many limitations that have prevented its widespread use. Most importantly, the moving image's orientation was completely ignored, so its axes were always assumed to be parallel to x, y and z. `itk::PatchedRayCastInterpolateImageFunction` solves the aforementioned problem by offering calculations of the rays' direction with all the images' orientation taken into account. The class interface was kept identical to `itk::RayCastInterpolateImageFunction`, thus the input image and focal point are set using the `SetInputImage` and `SetFocalPoint` methods.

However, not all limitations of `itk::RayCastInterpolateImageFunction` were addressed. Among the ones left unsolved were the lack of support for multi-threading and the bilinear scheme used for interpolation, which could be extended to tri-linear. These improvements were left for future work.

3 Example applications

Source code is included with this article and can be compiled on multiple platforms using CMake 2.8.1 or newer (`http://www.cmake.org`). Unit tests are provided for all the framework's classes, which can be executed using CMake's testing program, CTest. Unit testing requires a set of sample images, provided in the `Data` directory found within the source code's root directory.

In addition to the unit tests, two sample applications are provided: `MultiImageSearch` and `MultiImageRegistration`, which give an example of the framework's capabilities. Descriptions of these applications are given in the following sections, along with results with simulated data, that is, using DRRs as reference images instead of actual radiographs. The `MultiImageRegistration` is also tested with real images obtained from a public database.

3.1 Exhaustive search for intensity-based registration

The `MultiImageSearch` example is an example application that evaluates a chosen registration metric for various translation values using the `itk::ExhaustiveOptimizer`. The search is repeated on three resolution levels with downsampling factors of 4, 2 and 1 (1 meaning no downsampling). The results of this application should be plotted to examine the metric's shape around the optimum and design a suitable optimization strategy.

The `MultiImageSearch` command line looks as follows

```
MultiImageSearch [m] [N] [f1] [f1Px] [f1Py] [f1Pz] ... [fN] [fNPx] [fNPy] [fNPz]
  [mc] <sigma> [sx] [sy] [sz] [ds]
```

where m is the moving image's file name, N the number of fixed images, fk with $k = 1..N$ the k'th fixed image's file name and $fkPx$, $fkPy$ and $fkPz$ the real-world coordinates of the k'th image's focal point.

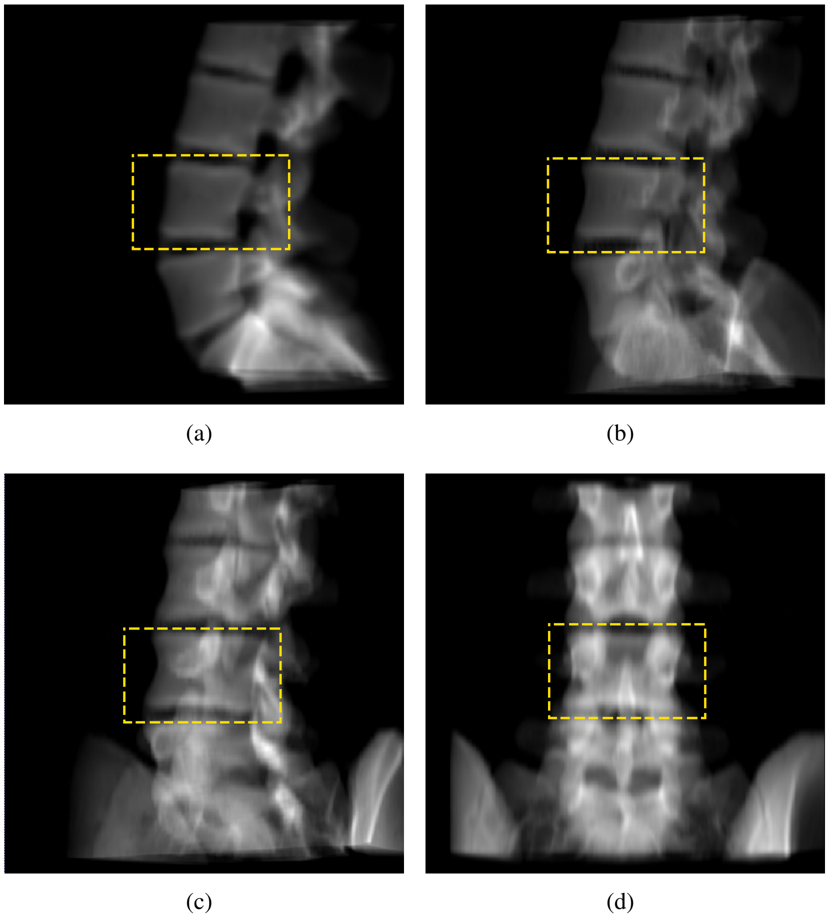


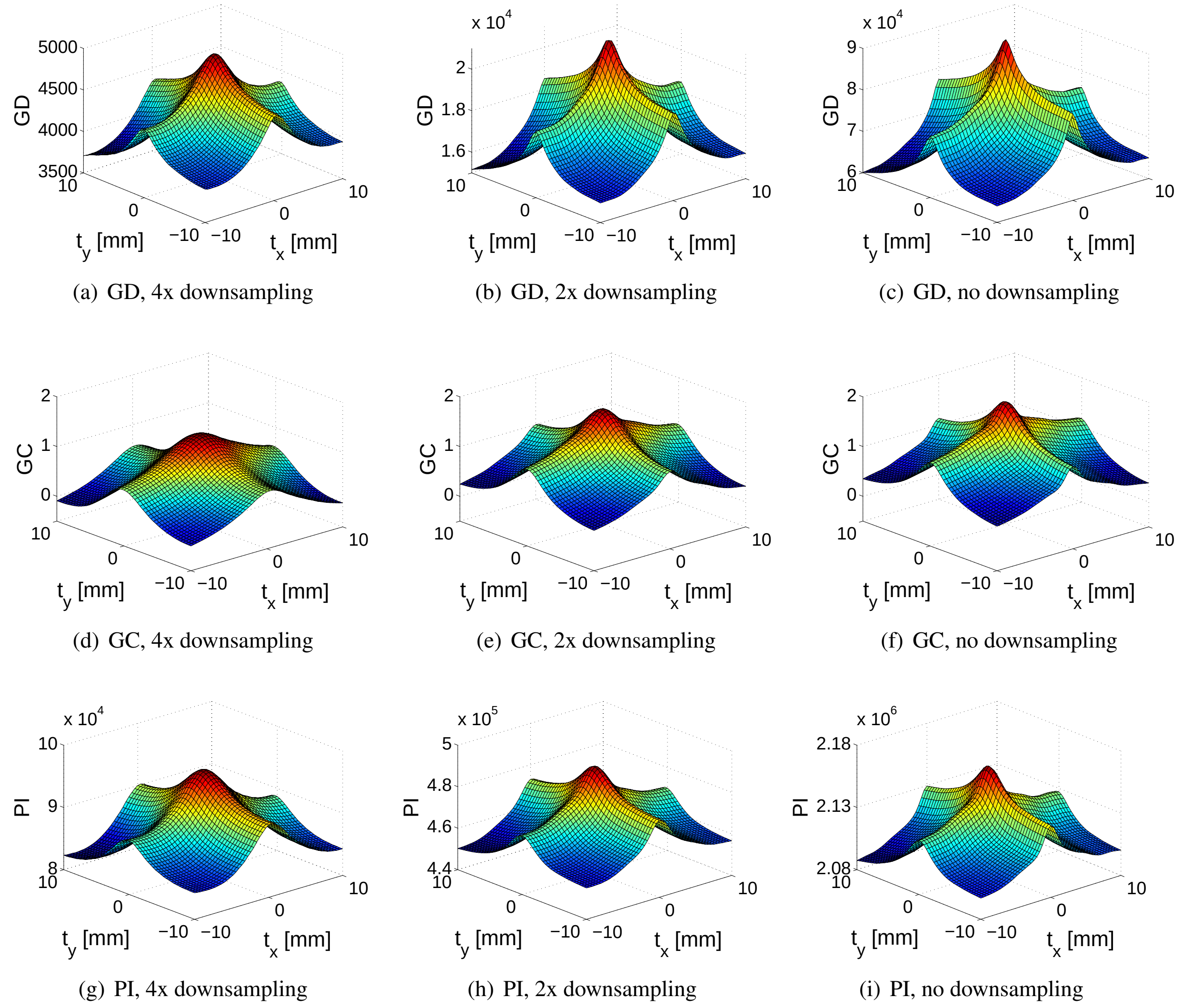
Figure 2: DRRs used as fixed images with their regions of interest (a) 0° LAT view (b) 30° (c) 60° (d) 90° AP view

Argument *mc* is the metric class, which can take values of *gd*, *gc* or *pi* for Gradient Difference, Normalized Gradient Correlation or Pattern Intensity respectively. Parameter *sigma* must only be given when the Pattern Intensity metric is chosen, and sets the σ value for Equation 10. Parameters *sx*, *sy* and *sz* define the search grid for each dimension, with the total number of steps being $2 \times s + 1$ centred around 0.0. Parameter *ds* controls the step size in millimetres.

Calling `MultiImageSearch` on the `Data` directory with the following parameters

```
MultiImageSearch moving.thr.mha  2 fixedAP.roi.mha 0 1000 0 fixedLAT.roi.mha -1000 0 0
  gd 25 25 0 0.4
```

evaluates the Gradient Difference metric around the optimum transform, which is the identity, on a 51 by 51 grid with different translation values on the x and y axes. The moving image `moving.thr.mha` is a CT scan of the spine lumbar section, thresholded for a more realistic generation of DRRs and located with the real-world origin inside the L4 vertebra. Images `fixedLAT.roi.mha` and `fixedAP.roi.mha`, shown in Figure 2(a) and 2(d), are lateral (LAT) and antero-posterior (AP) simulated projections of the CT dataset, cropped to show only vertebra L4. Focal point for the LAT projection is $(-1000, 0, 0)$ and, for the AP projection, $(0, 1000, 0)$. Changing argument *gd* to *gc* or *pi* (setting σ to 2000 when Pattern Intensity is chosen) makes the same experiment changing only the similarity metric. The resulting plots for all metrics on all resolution



(a) GD, 4x downsampling (b) GD, 2x downsampling (c) GD, no downsampling

(d) GC, 4x downsampling (e) GC, 2x downsampling (f) GC, no downsampling

(g) PI, 4x downsampling (h) PI, 2x downsampling (i) PI, no downsampling

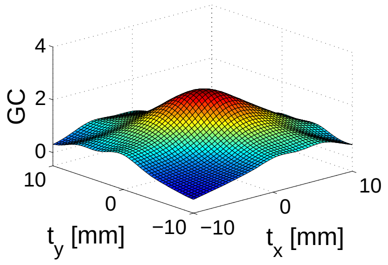
Figure 3: Metrics' plots around the optimum transform for different translation values. Tested metrics are Gradient Difference (top row), Normalized Gradient Correlation (middle row) and Pattern Intensity (bottom row). Resolution level changes across columns with downsampling factors of 4 (left column), 2 (middle column) and 1 (right column).

levels are shown in Figure 3. Note that the optimum is well-defined on all resolution levels and that narrow ridges are present along the x and y axes, which can slow down an optimizer's convergence. These ridges are effectively reduced at lower resolution levels.

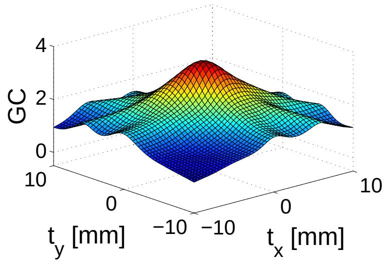
Extra images can be added into the registration by modifying the command line. For example, calling `MultiImageSearch` with the following arguments

```
MultiImageSearch moving.thr.mha 4 fixedLAT.roi.mha -1000 0 0  fixed30.roi.mha -866.0254
  500 0   fixed60.roi.mha -500 866.0254 0   fixedAP.roi.mha 0 1000 0   gc 25 25 0 0.4
```

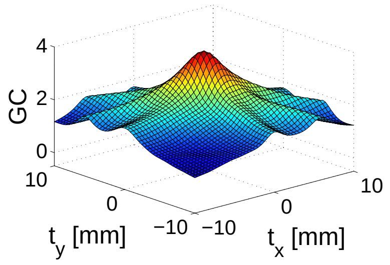
performs an exhaustive search of the Normalized Gradient Correlation metric, but using four fixed images instead of two. Apart from the LAT and AP views, corresponding to 0° and 90°, two extra views with angles of 30° and 60°, shown in Figure 2(b) and 2(c), were added into the metric calculation. Results shown on



(a)



(b)



(c)

Figure 4: Normalized Gradient Correlation plots around the optimum transform using 4 fixed images. Comparison with the metric using 2 fixed images (middle row of Figure 3) show that additional ridges appear on the normal directions of the added images, which are effectively filtered at low resolution levels.

Figure 4 depict the metric around the optimum for the three tested resolution levels. Note that additional ridges appear, which correspond to the normal directions of the added images.

3.2 Multi-resolution intensity-based registration

The `MultiImageRegistration` example performs a rigid registration of the moving image to the set of fixed images given as input. The Normalized Gradient Correlation metric is used to evaluate the images' similarity, which is optimised using the Fletcher-Reeves Polak-Ribiere (FRPR) Conjugate Gradient algorithm. Three resolution levels are used, with downsampling factors of 4 and 2 for the first and second. The images' original resolutions are used in the last level.

The `MultiImageRegistration` is called with a command line similar to the previous example

```
MultiImageRegistration [m] [N] [f1] [f1Px] [f1Py] [f1Pz] ... [fN] [fNPx] [fNPy] [fNPz]
  [oDir] <inT>
```

with the main difference being the last two arguments. The *oDir* argument is the name of the directory where the registration results should be stored and *inT* the input transform in the ITK's format. In the output directory, `MultiImageRegistration` will save the `outTransform.txt` file with the output transform, the `outMatrix.txt` text file with the same transform but written as a 4 by 4 matrix, the `inMatrix.txt` file with the input transform matrix and the moving image's projections named `projectionk.mha` with k from 0 to $N-1$. Lastly, subtraction between the projections and the fixed images are stored as `subtractionk.mha`.

Running `MultiImageRegistration` with the following parameters

```
MultiImageRegistration moving.thr.mha  2 fixedAP.roi.mha 0 1000 0 fixedLAT.roi.mha -1000 0 0
  outDir inTransform.txt
```

registers the CT scan of the previous examples to its DRRs taken at the LAT and AP orientations. Results are saved in the `outDir` directory and `inTransform.txt` is an example transform which serves as a starting point for the registration, with angles of -0.1, -0.2 and 0.2 radians for the x, y and z axes (-5.73°, -11.4592° and 11.4592° respectively) and displacements of -4.9, -4.7 and 0.7 millimetres along x, y and z. After registration, transform parameters are set to -0.0003, 0.0002 and 0.0003 radians (-0.0172°, 0.0115° and

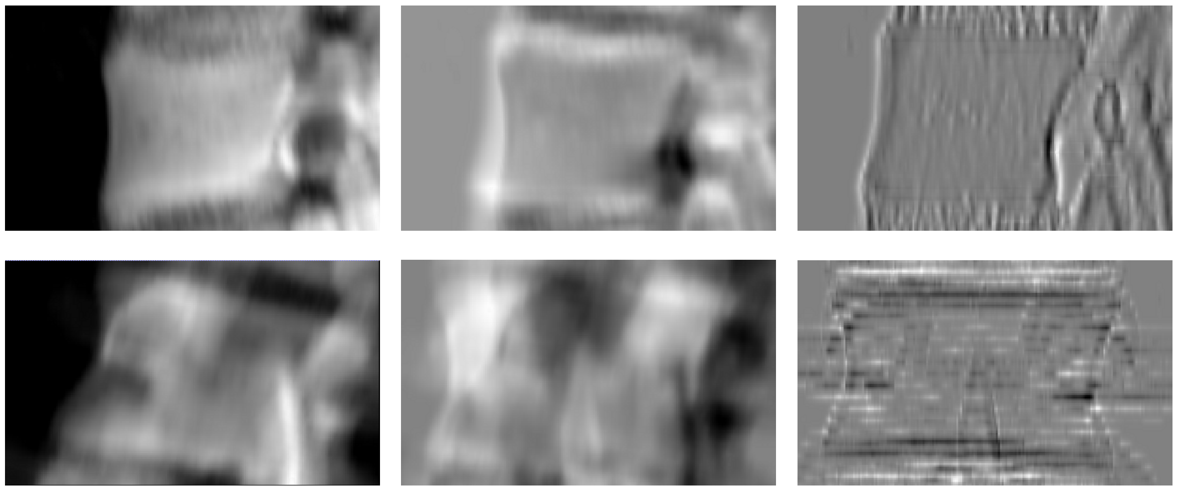
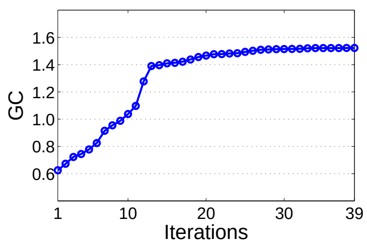
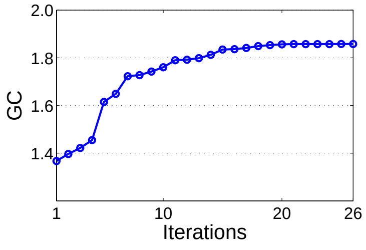


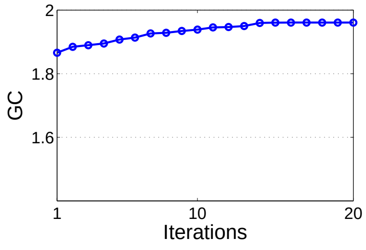
Figure 5: Results of the `MultiImageRegistration` program using a LAT (top row) and AP (bottom row) fixed images. Figures show the projections before registration (left column), the subtraction between each projection and its corresponding fixed image before registration (middle column) and the subtraction after registration (right column).



(a)



(b)



(c)

Figure 6: Normalized Gradient Correlation plots of the `MultiImageRegistration` example for all three resolution levels (a) 4x downsampling (b) 2x downsampling (c) no downsampling

0.0172° respectively) and -0.0029, 0.0168 and -0.0147 mm, which is quite close to the correct transform which has all its parameters equal to zero. Figure 5 shows the projections' aspect before registration and the subtractions between the projections before and after registration, which clearly show how matching between the images is improved. Figure 6 shows the Normalized Gradient Correlation metric evolution for the three resolution levels of the registration.

Testing of the `MultiImageRegistration` application with a more realistic input was done using the dataset provided by the Image Sciences Institute of the University of Utrecht (`http://www.isi.uu.nl/Research/Databases/GS/`), comprised of CT, MR, three-dimensional radiography (3DRX) and fluoroscopy images of eight different vertebrae and complemented with a standardised protocol for evaluation of registration algorithms [8]. The aforementioned protocol defines the ground-truth transform for each image and 200 initial transforms with mean Target Registration Errors (mTRE) uniformly distributed between 0 and 20 mm. Combination of the vertebrae and starting transforms give a total of 1600 registrations for evaluation of each algorithm, which are considered successful if their final mTRE is below 2 mm. The protocol also establishes calculation of the methods' capture range as the value of the initial mTRE with a percentage of successful registrations of 95%.

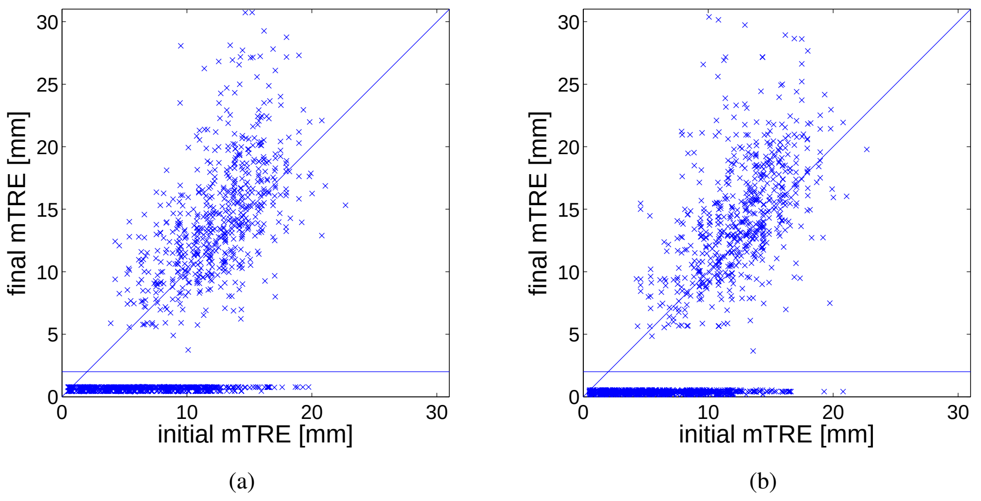


Figure 7: Standard evaluation of the `MultiImageRegistration` application using 2 x-ray images for registration of a 3DRX (a) and a CT (b) datasets. Each point represent a registration result on eight different vertebrae (200 registration per vertebra, 1600 in total). Registration are considered successful when their final mTRE is less than 2mm, indicated by the plots' horizontal line.

The `MultiImageRegistration` was evaluated with the 3DRX and CT images. The MR images were not used as their intensities have no direct relation with the x-rays', which prevents the use of intensity-based methods. Figure 7 shows the initial and final mTRE for each of the 1600 registrations performed for each imaging modality. For the 3DRX, mean mTRE of the successful registrations was 0.5827 mm, success rate was 62.06% and capture range was 6.6864 mm. For the CT images mean mTRE was 0.3511 mm, success rate was 61.31% and capture range was 7.5229 mm. Comparison of these values with evaluations of other algorithms under the same protocol [3] show that `MultiImageRegistration` fares well in comparison with intensity-based [4], gradient-based [7] and reconstruction-based methods [6], but requires additional robustness to reach the level of more sophisticated methods, such as the Robust Gradient Reconstruction-Based Extended method (RGRBe) proposed by Markelj et al [3].

Due to licensing reasons, it is not possible to include the used datasets with this article. However, they are available for public use on the Image Sciences Institute website (`http://www.isi.uu.nl/Research/Databases/GS/`) and interested readers are encouraged to download them for their own research.

4 Conclusions and future work

A framework for 2D-3D registration has been proposed which effectively increases the ITK capabilities for this type of application. The framework allows registration of a single moving 3D image to multiple fixed 2D projections, which solves the ITK's limit of a single fixed image per registration. In addition, both moving and fixed images can be defined in any position and orientation in space, which solves the problem of the moving image's orientation which, in the original ITK implementation, was completely ignored and assumed to be parallel to the world's axes. Two additional similarity metrics were added –Normalized Gradient Correlation and Pattern Intensity– and a new implementation of the Gradient Difference metric was proposed. The framework's interface has been kept similar to the one used by ITK, so any developer familiar with the toolkit should become accustomed to the proposed framework with minimal effort. Source code, unit tests and example applications are provided, along with results that demonstrate the framework's capabilities with real and simulated data.

Future work should address limitations such as the lack of support for multi-threading and the restriction for transforms, which are currently limited to rigid ones due to the ray-casting algorithm's implementation. In addition, only intensity based registration was addressed, which prevents the application of the framework in cases where a direct correlation between the intensities of the moving and fixed images does not exist, such as the case of registration of 3D MR images to multiple x-ray projections. For this, gradient-based methods could be used [2], which would require additional extensions to ITK.

Acknowledgements

The authors would like to thank Jian Wu, Aviv Hurvitz and Martjn van der Bom for their contributions to this work.

The 3DRX, CT, MR and X-ray images and ground truth for the 2D-3D registration used in this work were provided by the Image Sciences Institute, University Medical Center Utrecht, The Netherlands.

References

[1] L Lemieux, R Jagoe, D R Fish, N D Kitchen, and D G Thomas. A patient-to-computed-tomography image registration method based on digitally reconstructed radiographs. *Medical physics*, 21(11):1749–60, November 1994. 2.3

[2] P Markelj, D Tomaževič, B Likar, and F Pernuš. A review of 3D/2D registration methods for image-guided interventions. *Medical image analysis*, (in press), 2010. 1, 2.2, 4

[3] P Markelj, D Tomaževič, F Pernuš, and B Likar. Robust 3-D/2-D registration of CT and MR to X-ray images based on gradient reconstruction. *International Journal of Computer Assisted Radiology and Surgery*, 3(6):477–483, July 2008. 3.2

[4] G P Penney, J Weese, J A Little, P Desmedt, D L Hill, and D J Hawkes. A comparison of similarity measures for use in 2-D-3-D medical image registration. *IEEE transactions on medical imaging*, 17(4):586–95, August 1998. 2.3, 2.3, 3.2

[5] P Steininger, M Neuner, and R Schubert. An extended ITK-based Framework for Intensity-based 2D/3D-Registration, 2009. 1

[6] D Tomaževič, B Likar, and F Pernuš. 3-D/2-D registration by integrating 2-D information in 3-D. *IEEE transactions on medical imaging*, 25(1):17–27, January 2006. 3.2

[7] D Tomaževič, B Likar, T Slivnik, and F Pernuš. 3-D/2-D registration of CT and MR to X-ray images. *IEEE transactions on medical imaging*, 22(11):1407–16, November 2003. 3.2

[8] E B Van De Kraats, G P Penney, D Tomazevic, T Van Walsum, and W J Niessen. Standardized evaluation methodology for 2-D-3-D registration. *IEEE Transactions on Medical Imaging*, 24(9):1177–1189, 2005. 3.2

[9] 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. In *Proceedings of SPIE*, volume 7962. Society of Photo-Optical Instrumentation Engineers (SPIE), March 2011. 2.4

[10] J Weese, T M Buzug, C Lorenz, and C Fassnacht. An Approach to 2D / 3D Registration of a Vertebra in 2D X-ray Fluoroscopies with 3D CT Images. *Lecture Notes in Computer Science*, 1205:119–128, 1997. 2.3