# A Correspondence-Based Software Toolkit for Image Registration

Chia-Ling Tsai, Charles V. Stewart, Amitha Perera, Ying-Lin Lee, Gehua Yang, and Michal Sofka

***Abstract*— This paper presents a correspondence-based toolkit for image registration. Written in C++, the toolkit complements the capabilities of the Insight Toolkit (ITK). Major components include features, feature sets, match generators, error scale estimators, robust transformation estimators, and convergence testers, all combined and controlled by several different registration engines. Correspondence-based algorithms which can be implemented using the toolkit extend from ICP to hybrids of intensity-based and feature-based registration. The toolkit is being used both as an education tool and the foundation for developing new algorithms.**

## I. Introduction

This paper presents a software architecture for correspondence-based registration algorithms, RGRL (Rensselaer Generalized Registration Library), and demonstrates several example algorithms and applications. The notion of a correspondence extends from traditional closest-point based matching [1] to recent hybrids of intensity-based and feature-based techniques [3], [12], [20] where the matching of image intensities creates temporary correspondences. Correspondences, however they are generated, are used to refine the transformation estimate. RGRL is written in C++ and built on top of the VXL libraries [22] which were designed for computer vision applications. The library is actively maintained, and the most part which is well tested is now open-source for the purpose of research and education [15]. The remaining will be released soon.

For image registration, there are two other well-developed and widely-known libraries/toolkits maintained by the open-source communities: OpenCV [2] for computer vision and ITK [8] for medical imaging. OpenCV is a collection of algorithms for various computer vision problems with the focus on real-time computing, such as human-computer interaction, recognition, and motion tracking. ITK contains techniques for medical imaging processing, segmentation, and registration. The focus of the ITK registration framework is intensity-based methods. Our approach complements that of ITK by emphasizing correspondence-based registration. Although we would eventually like to fully-integrate with ITK[1], our library currently compiles with ITK, allowing components of each toolkit to be used in any program.

We are only aware of few other efforts at building open-source software for image registration, primarily using visualization packages. VTK-CISG [6] and SLICER [4] implement registration based on maximizing Normalized Mutual Information [21] for alignment of 3D medical images. Scanalyze [13] is for 3D range data using Iterative Closest Point (ICP) algorithm.

A preliminary version of the RGRL design approach was presented in [20]. The current paper expands on this design and describes the software architecture and its properties. The RGRL toolkit implements a general framework for correspondence-based registration. Components of the framework include initializers, feature sets, correspondence generators, robust transformation estimators, and convergence testers. Multiple feature types may be used either simultaneously or sequentially, and multiresolution is handled naturally. The role of each component is isolated by being designed as an abstract base class, with a wide-variety of techniques being implemented as derived classes. Variations on the general framework, including adaptive transformation model selection and registration region growing [16], are facilitated by the notion of a *view*, first introduced in [20]. The overall design goal of the library is to combine simplicity for education on image registration and flexibility for development of new registration techniques that fall within the general class of correspondence-based approaches.

## II. General Correspondence-Based Architecture

The framework has an inner flow loop (see Fig. 1) implementing a basic but flexible correspondence framework and a set of outer loops controlled by a "registration engine" that allow for multiple initial estimates, multiple feature types, and multiple stages (resolutions) to the registration process. Together these realize a flexible, generalized correspondence-based architecture. There is a clean separation between computational objects and representation objects. The computational objects take one or more representation objects as input and produce a representation object as output. Each component can be removed from the framework and used separately. For example, the match set could be used as part of a random-sampling search rather than a reweighted least-squares framework. The remainder of this section describes

All work occurred while authors worked at Rensselaer Polytechnic Institute

Chia-Ling Tsai is with Faculty of Department of Computer Science and Information Engineering, National Chung Cheng University, Chiayi 62102, Taiwan (e-mail: tsaic@cs.ccu.edu.tw)

Charles V. is with Faculty of Department of Computer Science, Rensselaer Polytechnic Institute, Troy, NY 12180, USA (e-mail: stewart@cs.rpi.edu)

Amitha Perera and Ying-Lin Lee are with GE Global Research, Niskayuna, NY, USA (e-mail: {perera,leey}@cs.rpi.edu)

Gehua Yang and Michal Sofka are with Department of Computer Science, Rensselaer Polytechnic Institute, Troy, NY 12180, USA (e-mail: {yangg2,sofka}@cs.rpi.edu)

[1]Our library was not built on top of ITK because several underlying software components pre-date ITK.

the architecture in more detail: the first few subsections summarize the core components of the inner loop, and the last two subsections describe the registration engine and the view-based architecture.

### A. Features, Feature Sets and Error Projectors

A *feature* is the basic element for correspondence-based registration. It can be as simple as a point in space with a location, or as complex as a region with intensity information and orientation information encapsulated. Features can be constructed before matching is started, or in the case of matching intensities (block matching) (see for example [12]), constructed as part of the matching process. To construct features prior to registration, one may implement a feature extractor using ITK or other VXL libraries and store the features in a pre-defined ASCII format for the *feature reader* to import the features.

Different feature types should define different alignment error. For example, the alignment error of keypoint-to-keypoint correspondences is best described as a Euclidean distance, since keypoints are fairly stable point locations. For "non-distinctive" points on curves and surfaces (e.g. blood vessel), however, this is too restrictive a measure because the features are often poorly localized in the direction along the curve or on the surface. Thus, the error in the direction normal to the curve or surface is the most appropriate error distance measure.

To uniformly compute an error distance for different feature types, we pre-multiply the standard Euclidean error vector by an *error projector matrix*, and then compute the magnitude of the projected vector. To better illustrate the concept, let $\mathbf{p}$ be a moving image feature, $\mathbf{q}$ be its corresponding fixed image feature, $\mathbf{T}$ be the transformation with parameters $\mathbf{\Theta}$, and $\mathbf{d} = \mathbf{T}(\mathbf{p};\mathbf{\Theta}) - \mathbf{q}$ be the error vector. Then, if $\mathbf{q}$ is a point on a surface, with local normal $\hat{\boldsymbol{\eta}}$, the error measure should be the normal distance $|\mathbf{d}^T\hat{\boldsymbol{\eta}}|$. By contrast, if $\mathbf{q}$ is a point on a curve with tangent vector $\hat{\mathbf{t}}$, the error measure should be the distance to the closest point on a linear approximation to the contour at $\mathbf{q}$: $\|\mathbf{d} - [\mathbf{d}^T\hat{\mathbf{t}}]\hat{\mathbf{t}}\|$. With the error projector, $\mathbf{M}$, the square alignment error is

$$(\mathbf{T}(\mathbf{p};\mathbf{\Theta}) - \mathbf{q})^T\mathbf{M}(\mathbf{T}(\mathbf{p};\mathbf{\Theta}) - \mathbf{q})$$

Error projectors of different error types are summarized in Table I. Section II-D explains their importance in estimation.

Each feature type is associated with a *signature*, which encodes the geometric property of a feature. For instance, the resolution scale at which the feature is detected is part of the signature. For point-on-curve/point-on-surface, the tangent/normal direction is also included. Use of the signature provides additional information for the similarity of two matching features, which can serve as a weight for the correspondence. Taking point-on-surface for instance, let $(s_q, \hat{\boldsymbol{\eta}}_q)$ and $(s_p, \hat{\boldsymbol{\eta}}_p)$ be the scale-direction signatures for $\mathbf{q}$ and $\mathbf{T}(\mathbf{p};\mathbf{\Theta})$, respectively. Assuming $s_q \geq s_p$, the signature similarity for a point-on-surface correspondence is defined as $(s_p/s_q)|\hat{\boldsymbol{\eta}}_q^T\hat{\boldsymbol{\eta}}_p|$. Different signature properties can be easily handled by deriving from existing feature types.

All features of the same type from one image are stored in (or generated from) a *feature set*. Fundamentally, feature sets answer queries in support of correspondence generation. Examples include finding the nearest feature in the fixed image to a given feature (mapped from the moving image), finding the k-nearest neighbors, or even finding all features in a given region when multiple correspondences are required for a feature (see for example [5]). Spatial data structures such as k-d trees and digital distance maps support fast querying. A specialized feature set may store an image and generate features by finding matches using intensity comparisons over a given region.

### B. Initialization

The *initializer* provides the first set of matches required at the start of the iterative process illustrated in Fig 1. The interface to the initializer object defines only the request to provide the next initial estimate, which is passed to the main body of the registration engine for refinement. This isolates initialization from the rest of the algorithm. Besides the initializer with a prior transformation, a number of common initialization techniques have been implemented as the derived classes:

- **Random-sampling initializer:** Random-sampling is performed on a pre-computed match set. The initializer returns only one transformation, which is the best estimate out of the random-sampling process.
- **Invariant-indexing initializer:** The initializer generates a list of landmark/keypoint matches by comparing feature descriptors invariant to some low-order transformation models, for example, similarity [9], [16]. The matches are ranked in descending order, and each match provides an initial transformation. At each request for the next initial estimate, the next available match in the list is returned for refinement.
- **Moments-based initializer:** The initializer returns a similarity transformation that best aligns the first and second moments of the two point sets from the image pair.
- **Gradient-based initializer:** Building on top of the invariant-indexing initializer, the transformation is refined using images of directional gradient magnitudes and the steepest-descent minimization technique.

TABLE I

ERROR PROJECTOR MATRIX FOR DIFFERENT TYPES OF ERRORS. FOR A POINT ON A CURVE, $\hat{\mathbf{t}}$ IS THE TANGENT TO THE CURVE, AND FOR POINT ON A SURFACE, $\hat{\boldsymbol{\eta}}$ IS THE NORMAL TO THE SURFACE. $m$ IS THE NUMBER OF DIMENSIONS OF THE IMAGE. ALL ERROR TYPES ARE APPLICABLE IN 2D AND 3D, BUT POINT-TO-CURVE AND POINT-TO-SURFACE ARE ONLY DIFFERENTIABLE IN 3D.

| **Error Type** | point-to-point | point-to-curve | point-to-surface |
|---|---|---|---|
| **Error Projector** | $\mathbf{M} = \mathbf{I}$ | $\mathbf{M} = \mathbf{I} - \hat{\mathbf{t}}\hat{\mathbf{t}}^T$ | $\mathbf{M} = \hat{\boldsymbol{\eta}}\hat{\boldsymbol{\eta}}^T$ |
| **DoF to Constraint** | m | m-1 | 1 |

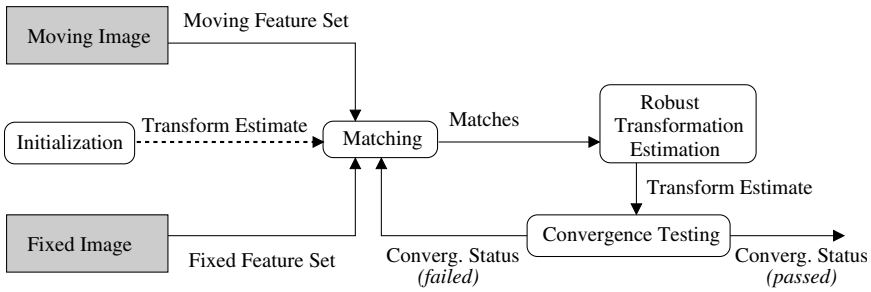


Fig. 1. Architecture of the inner loop of general correspondence-based registration. Each computational component, enclosed in the box, takes one or more representation objects as input, and produces one representation object as output, indicated by the arrow. All representation objects constructed stay in the loop until re-computed. The dashed arrow indicates the flow of object which is only used once for initialization.

Other initializers, such as coarse sampling of a small parameter space, could be easily added to the library in the future.

### C. Matching

The *matcher* is responsible for computing the correspondences and constructing a *match set* data structure to store them. Two matching techniques are implemented (as derived classes):

- **K-nearest matcher:** This matcher can use different "nearness" measures, such as Euclidean distance and signature-based similarity measures [3], [14], to determine the $k$ nearest neighbours (allowing multiple correspondence per feature). When $k = 1$ and "nearness" is defined by Euclidean distance, this becomes the classical ICP matcher.
- **Block matcher:** This is an algorithm that creates a temporary correspondence (and feature) by taking a small region centered on a feature in the moving image and finding the best corresponding region in the fixed image based on an intensity match.

### D. Transformation Objects and Estimators

The transformation object represents the current estimate of the transformation and its uncertainty. It provides the ability to map image locations and direction vectors (approximately for non-rigid transformations). Transformation models currently implemented include translation, rigid, similarity, affine, quadratic, hierarchical B-splines, homography, and homography with radial lens distortion.

For each transformation model there is an associated estimator. Each takes a set of matches and constructs a transformation estimate. This estimate generally includes the covariance matrix of the estimated transformation parameter vector.

All estimators share two important properties:

- Each estimator computes a weighted (non-linear) least-squares estimate of the transformation. Centering and normalization are used to increase numerical stability.
- Constraints are constructed using the error projector (Table I). Therefore, the estimators need not know anything about the feature types forming the matches. This means new feature types (there are five currently) may be added without having to change the estimators.

### E. Robust Estimation

Robust estimation must be incorporated into the toolkit to avoid the inevitable incorrect matches. M-estimators are used in RGRL, implemented using iteratively reweighted least-squares [17]. This requires weighting functions and robust estimation of the standard deviation of the errors (the error "scale"). *Weighter* objects compute the weights for each match and store them back with the matches in the match set. Several types of weight calculations are implemented, including standard M-estimator weighting, weighting based on feature similarities, and competitive weighting, if multiple matches for each feature. Error scale estimation is implemented in two ways: weighted techniques compute simply the (robustly) weighted average of the square errors; unweighted techniques use the MUSE algorithm [11], which automatically determines the fraction of "inlier" matches. The error scale determines the correctness of the estimation.

### F. Convergence Testing

A *convergence tester* determines whether the registration process should terminate for the current initial estimate. The result of testing is stored in a *convergence status* object. Three termination criteria are currently implemented: (1) the estimation has converged with the change of alignment error below certain threshold, (2) the estimation is oscillating, and (3) the stability, which is the inverse of the largest transfer error [7, Ch. 3] of the sampled points, is above a threshold.

Both weighted and unweighted convergence testers are implemented as derived classes. The alignment error computed by the weighted tester is the weighted average of the errors, while the unweighted tester returns the median error.

### G. The Registration Engine

The framework is encapsulated in the *registration engine*, which automatically handles multiple initial estimates, multiple feature types, and multiple stages:

- **Multiple initial estimates:** The engine loops through the set of initial estimates from the initializer. If any of the initial estimate leads to a satisfactory final estimate, the engine terminates immediately. If no estimate satisfies the termination criteria, the overall best estimate is returned.
- **Multiple feature types:** The estimate of the transformation can be computed from multiple feature sets using

different feature types. An example would be combination of fiducial and surface points for registration of CT images of the head [10]. Each feature set pair is accompanied by a matcher, a weighter and a scale estimator.

- **Multiple stages:** A *stage* contains the transformation model(s), and a set of feature set pairs which are used together to estimate a single transformation. The transformation model(s) can differ between stages, and the final estimate from one stage is used as the initial estimate in the next. This is used to realize multiresolution feature-based registration.

The flow-diagram in Fig 1 above illustrates the simplest scenario: one initial estimate, one feature set pair and one stage.

### H. View-Based Architecture

Another level of flexibility is achieved by allowing view generation, as illustrated in Fig 2. A *view* is a snapshot of the status of the registration process. It is a combination of a stage (resolution), a transformation model and estimate, and an image region over which the model currently applies. The view-based framework unifies a number of variations of general correspondence-based registration.

Through the manipulation of the *view*, the *view generator* may define complex behavior during registration. We have implemented a class of bootstrap view generators [16], [24], which applies the uncertainty in the transformation estimates to automatically control either or both region growth and model selection throughout the registration. Different model selection criteria can be plugged into the view generator. Other variations of correspondence-based registration can be built into the view-based framework in the future by defining new view generators.

## III. Algorithms and Applications

Well-documented tutorial examples are provided with the library, following the style of the ITK software guide [8]. Further examples will be provided as the toolkit matures and the application base widens. This section presents four example algorithms[2], each with an application fully-implemented using the toolkit:

- **Iterative Closest Point (ICP):** Standard ICP [1] is implemented using a single stage and matching using a k-nearest neighbor matcher with $k = 1$. By simply setting different feature types the toolkit switches between registration of 3D neurons (point-on-curve) and registration of range images (point-on-surface).
- **Dual-Bootstrap Iterative Closest Point (DB-ICP):** We re-implemented DB-ICP algorithm [16], [19] using the toolkit. DB-ICP registers retinal fundus and fluorescein images (Fig 3). As alluded to above, the initializer automatically generates possible vascular landmark correspondences by matching invariant signatures. Each correspondence is used to initialize a view that contains a low-order transformation and a small image region. Dual-bootstrap view generation, controlling ICP registration, grows initial and refines estimates to obtain final estimates. For the retina application, the final model is quadratic.
- **Generalized Dual-Bootstrap Iterative Closest Point (GDB-ICP):** Extending DB-ICP, we developed a system capable of aligning image pairs having some combination of low overlap, substantial orientation and scale differences, substantial illumination changes, substantial scene changes, and different modalities [24]. With the DB-ICP framework in place, we achieved generalization by doing generic feature extraction, nonlinear estimation of homography (with lens distortion), and sophisticated acceptance criteria based on accuracy, stability and consistency of the matched features. The generic features are multi-scale corner and face points. The invariant-indexing initializer generates rank-ordered Lowe's keypoint [9] matches based on SIFT descriptors. The initial transformation is refined in DB-ICP fashion using multiple feature types, corner and face points, simultaneously. An example mosaic of images with low overlap and illumination changes are shown in Fig 4. The executable is available online at [23].
- **Iterative Most Similar Point (IMSP):** We developed an algorithm for hybrid feature-based and intensity-based registration. Feature sets from the moving image are matched to the intensity structure of the fixed image, generating temporary features and correspondences. Similarity-based weighting is used and the uncertainty in the alignment is used to control the search range for correspondences. Multiple resolution stages are used as well. Details are in [18]. An example alignment of serial lung CT images in B-spline deformable registration is shown in Fig 5.

[2]Some components allow parameter tuning for better efficiency, not robustness. However, we tested each of the 4 example algorithms with a fixed set of parameters.

## IV. Conclusion and Discussion

This paper presented a software toolkit that realizes a generalized architecture for correspondence-based registration. The toolkit is designed for two levels of interaction. The first level consists of the registration engines and sample applications. It is targeted at both the practitioner who wants to just use the various registration algorithms and at the researcher who wants to experiment with new techniques within the framework of iterative correspondence-based registration algorithm by specializing one or more of the objects. The second level consists of the independent components, which can be used to develop new registration engines and applications, expanding the scope of research questions to which RGRL may be applied.

Well-documented tutorial examples are provided with the library, starting with the simplest form of registration, and proceeding slowly up and including view-based registration. We built these following the example of ITK [8] and have found them to be extremely useful as a teaching tool in our

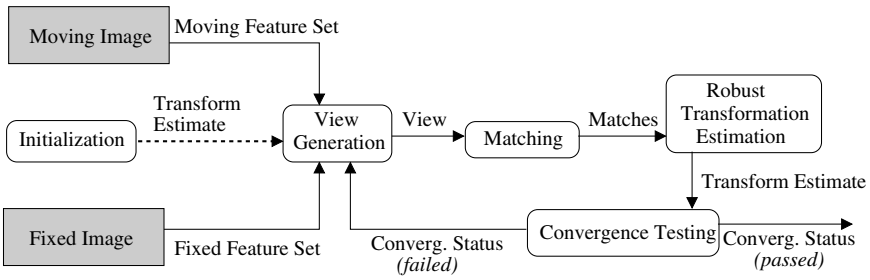


Fig. 2. Architecture of the inner loop of view-based registration. *View generation* defines complex behavior during registration, such as growth of region for registration and selection of the transformation model which is most appropriate for the the current *view*.

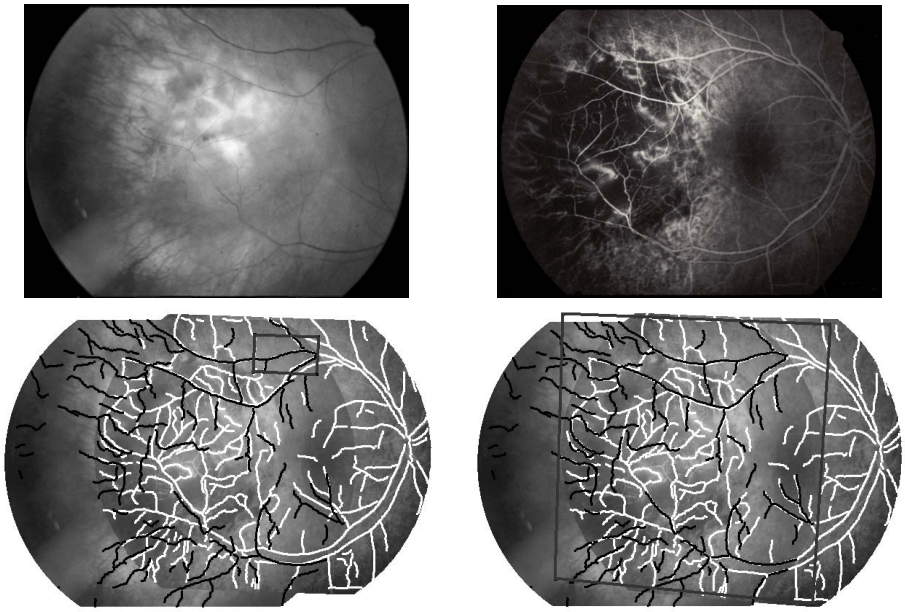

Fig. 3. Multimodal retinal image registration of a diseased eye using DB-ICP technique. The top row contains the red-free (left) and the fluorescein angiogram (right) images. Bottom-left: The initial alignment of the two images with a similarity transformation which is only good in the initial bootstrap region, shown as the grey box. Vessels extracted are shown in black on the red-free image and white on the fluorescein angiogram image. Bottom-right: The final alignment with sub-pixel accuracy using a quadratic model. The bootstrap region covers the entire overlap region between the two images.

courses. Further examples will be provided as the toolkit matures and the application base widens.

RGRL complements ITK, which focuses on intensity-based registration. Our eventual goal is to fully integrate RGRL with ITK to expand its dual-purpose role in teaching and research.

## REFERENCES


[1] P. Besl and N. McKay. A method for registration of 3-d shapes. *IEEE T. Pattern Anal.*, 14(2):239–256, 1992.

[2] G. Bradski. The openCV library. *Dr. Dobb's*, November 2000.

[3] J. Feldmar and N. Ayache. Rigid, affine and locally affine registration of free-form surfaces. *IJCV*, 18(2):99–119, 1996.

[4] D. T. Gering, A. Nabavi, R. Kikinis, N. Hata, L. J. O'Donnell, W. E. L. Grimson, F. A. Jolesz, P. M. Black, and W. M. Wells III. An integrated visualization system for surgical planning and guidance using image fusion and an open MR. *J. Magn. Reson. Imaging*, 13(6):967–975, June 2001.

[5] S. Granger and X. Pennec. Multi-scale EM-ICP: A fast and robust approach for surface registration. In *Proc. Seventh ECCV*, pages 418–432, 2002.

[6] T. Hartkens, D. Rueckert, J. Schnabel, D. Hawkes, and D. Hill. VTK CISG registration toolkit: An open source software package for affine and non-rigid registration of single- and multimodal 3d images. In *BVM2002, Leipzig, Springer-Verlag*, March 2002.

[7] R. Hartley and A. Zisserman. *Multiple View Geometry*. Cambridge University Press, 2000.

[8] L. Ibáñez, W. Schroeder, L. Ng, and J. Cates. *The ITK Software Guide: The Insight Segmentation and Registration Toolkit (version 1.4)*. Kitware Inc., 2003.

[9] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *IJCV*, 60(2):91–110, November 2004.

[10] C. R. Maurer, Jr., R. J. Maciunas, and j. Michael Fitzpatrick. Registration of head CT images to physical space using a weighted combination of points and surfaces. *IEEE Trans. Med. Imaging.*, 17(5):753–761, 1998.

[11] J. V. Miller and C. V. Stewart. MUSE: Robust surface fitting using unbiased scale estimates. In *Proc. CVPR*, pages 300–306, 18–20 June 1996.

[12] S. Ourselin, A. Rocher, G. Subsol, X. Pennec, and N. Ayache. Reconstructing a 3D structure from serial histological sections. *IVC*, 19:25–31, 2000.

[13] K. Pulli and M. Ginzton. Scanalyze: a system for aligning and merging range data. http://graphics.stanford.edu/software/scanalyze/.

[14] C. Schutz, T. Jost, and H. Hugli. Multi-feature matching algorithm for free-form 3d surface registration. In *ICPR98*, pages 16–20, 1998.

[15] D. Shelton, G. Stetten, S. Aylward, L. Ibanez, and C. Stewart. Teaching medical image analysis with the insight toolkit. *Med. Image Anal.*, submitted.

[16] C. Stewart, C.-L. Tsai, and B. Roysam. The dual-bootstrap iterative closest point algorithm with application to retinal image registration. *IEEE Trans. Med. Imaging.*, 22(11):1379–1394, 2003.

Fig. 4. Registration of a challenging image pair using GDB-ICP technique. The top row contains two images which were taken of Brugge Tower. Challenges for this image pair include illumination changes, and low overlap. The bottom image is a checker board image of the alignment result. The entire process is fully automatic, and the final alignment has sub-pixel accuracy using the transformation model of homography with radial lens distortion.

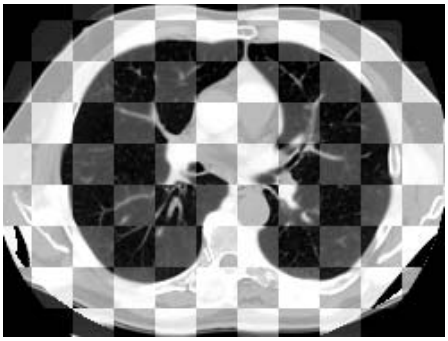

Fig. 5. An example of B-spline based registration, using IMSP, of 2 intra-patient 3D CT volumes taken 4 months apart. The figure shows checkerboard results (alternating blocks from the two images) in a single slice following alignment.


[17] C. V. Stewart. Robust parameter estimation in computer vision. *SIAM Rev.*, 41(3):513–537, 1999.

[18] C. V. Stewart, Y.-L. Lee, and C.-L. Tsai. An uncertainty-driven hybrid of intensity-based and feature-based registration with application to retinal and lung CT images. In *Proc. 7th MICCAI*, Saint-Malo, France, 2004.

[19] C. V. Stewart and B. Roysam. RIVERS: Retinal Image Vessel Extraction and Registration System. http://sgilinux4.aisl.rpi.edu/retina/.

[20] C. V. Stewart, C.-L. Tsai, and A. Perera. A view-based approach to registration: Theory and application to vascular image registration. In *Proc. IPMI*, Ambleside, UK, 20–25July 2003.

[21] C. Studholme, D. Hill, and D. Hawkes. An overlap invariant entropy measure of 3D medical image alignment. *Patt. Recog.*, 32:71–86, 1999.

[22] The TargetJr Consortium. *The VXL Book*. http://vxl.sourceforge.net/, 2000.

[23] G. Yang, C. V. Stewart, M. Sofka, and C.-L. Tsai. Generalized Dual Bootstrap-ICP Software Guide. http://www.cs.rpi.edu/research/groups/vision/gdbicp/exec/.

[24] G. Yang, C. V. Stewart, M. Sofka, and C.-L. Tsai. Alignment of challenging image pairs: Refinement and region growing starting from a single keypoint correspondence. *IEEE T. Pattern Anal.*, 2005.