Slice-wise, Non-rigid Volumetric Image Registration by Dynamic Programming

Eric R. Cosman

6.866 Final Project

We present a fast method for calculating a special class of non-rigid deformation between two volumetric images of similar physical structures. The method aligns volumes on a slice-wise basis by optimization of a measure of total slice-wise similarity. For a particular selection of slice directions in the volumes to be registered, arbitrary intra-slice deformation and slice thickness scaling are allowed. We use dynamic programming to find a globally optimal matching of slices, such that slice ordering is preserved.

This method is proposed as one for generating an initial transformation to initialize more general non-rigid volume matching algorithms. It has the advantage of being fast, able to capture large deformations, and always returns a transform with a Jacobian matrix everywhere positive definite (diffeomorphism). It is limited in that it relies on proper choice of the slice direction of each volume, and can only well-characterize a limited class for deformations, namely those in which slices move together. That is, where (x, y, z) parameterize one volume and (u, v, w) parameterized the other, and z and w are their respective slice directions:

$$\begin{pmatrix} u \\ v \\ w \end{pmatrix} = \begin{pmatrix} u(x,y,z) \\ v(x,y,z) \\ w(z) \end{pmatrix} \tag{1}$$

1 Context

Volumetric imaging techniques, which can provide a 3d sampling of *in vivo* tissue properties, are widely used in medicine for the purpose of diagnosis and surgical guidance. Often doctors and medical researchers would like to use images of the same physical structures in unision. For instance, in many circumstances, multiple images of an patient are taken at different times and with different imaging modalities (such as CT, MR, and PET) so that doctors may benefit from the complementary structural or functional information different images might provide. Furthermore, for the purposes of cross-sectional studies and building anatomical atlases, we want to combine images of the same anatomical region in different subjects. As a result, the problem of image alignment/registration is central to medical image analysis. The registration of images taken with different aparatuses, with patients in different positions, or of different subjects altogether, is complicated by the fact that the transformation between corresponding structures in these very different images is typically non-rigid.

Methods for finding such non-rigid deformations between volumetric images is an area of current medical vision research[1]. Typical methods combine some discretation of space (voxels, tetrahedra, surface elements), a smoothing term/physical model (elasticity, viscosity), and an image matching term (correlation, mutual information, etc.) to drive an iterative optimization whose endpoint is hopefully a good alignment. Running time and accuracy are sensitive to the deformation from which the iteration is initialized. Typical methods are computationally intensive, requiring 30 minutes to an hour to converge on high-performance workstations or computing clusters. Moreover, many techniques fail when deformations are large due to local minima in their optimization function. Even when multi-resolution approaches are used, some of these approaches fail because they do not produce valid deformations at low resolution.

2 Overview of Method

The motivation for the method presented here is to help overcome the limitations of current non-rigid alignment algorithms by finding large deformations that could initialize more refined alignment algorithms. Therefore, we'd like our method to operate efficiently on high resolution data while avoiding local minima in our matching function. To this end, we settle for alignment of 3d data

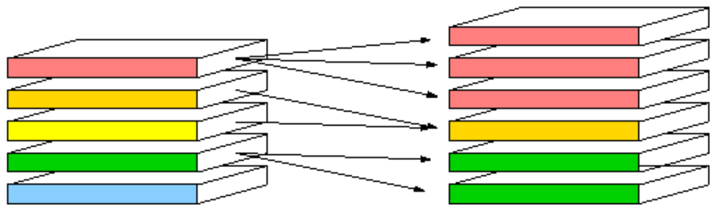


Fig. 1. Slice matching between two volumes. To model slice compression and expansion the matching function is multivalued in both directions. It may be that some slices in one volume will not map to any slices in the other.

on a slice-wise basis. That is, we cut each data cube into slices (resampling if necessary) and then match slices between volumes such that the slice order is preserved and some measure of similarity between the volumes is maximized.

[1] The following observations are from a conversation with Samson Timoner, who is currently doing research in this area at the AI lab and Brigham and Women's Hospital.

Furthermore, we allow the matching between slices to be multivalued in both directions to allow for slice compression and expansion. This process is illustrated in Figure 1.

We chose to do alignment by slice matching, because the 1d ordering of slices allows us to use dynamic programming[1] to efficiently search over all possible matchings that preserve slice ordering. As long as we formulate our objective function to treat slice pairings independently, the matching problem will have the optimal substructure required for use of dynamic programming. Since the resulting sequence alignment will be globally optimal (in some sense), we hope that would be able to recover large deformations even when working on high resolution images.

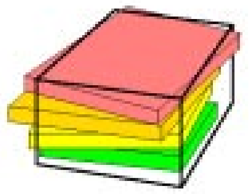


Fig. 2. Intra-slice deformation varies on a slice-wise basis. In this work we restrict the intra-slice deformation to alignment of second moments, i.e. 5 DOF, affine with rotation and stretching in the direction of each principle axis)

This framework does not constrain the manner in which we evaluate slice pairings. In fact, we could allow for an arbitary deformation to take place in considering pairs of slices, and this deformation can vary across slice pairings. Since we're only concerned with getting a rough estimate of the deformation and would like to do so quickly, we restrict the intra-slice deformation to alignment of second moments (affine) and use a sampled, intensity-based measure of similarity (such as the sample correlation or mutual information of corresponding pixel intensities). Figure 2 illustrates how the intra-slice deformation might vary across slices (without scaling slice thicknesses). To align the colored data slices to the data in the outlined volume, the colored sliced might be shifted, rotated and scaled independently[2].

Once an optimal matching between volumes has been evaluated, we have a mapping between the volumes' data coordinate, and we'd like to use it to initialize a non-rigid registration method. Say we want to warp a tetrahedral mesh on

[2] This independence could be viewed as a disadvantage if it lead to grossly different intra-slice transforms in adjacent slice pairing. If necessary, by a slight modification of the dynamic programming framework, we could basis our algorithm toward similarity between the intra-slice transforms of adjacent slice pairing in the matching.

one volume to another. It's clear how to use the in-slice affine deformation $\boldsymbol{A}_z, \boldsymbol{t}_z$ to adjust the verticies of the mesh: given the data slice index $z_0 = \text{round}(z)$ in which a vertex at (x, y, z) resides, set it's intra-slice coordinates (u, v, w) in the target volume as

$$\begin{pmatrix} u \\ v \end{pmatrix} = \boldsymbol{A}_{z_0} \begin{pmatrix} x \\ y \end{pmatrix} + \boldsymbol{t}_{z_0} \tag{2}$$

However, there is some ambiguity as to how to transform the slice coordinates when a slice maps to more than one slice in the other volume (e.g. the pink slice in Figure 2), and when a slice is not the sole target of a slice in the other volume (e.g the yellow and orange slices). For the former case, the simple solution which scales tetrahedra properly (positive Jacobian determinant) even when its verticies happen to fall within the same slice, is to scale the slice thickness by the multiplicity of its image under the matching, about the mean slice index of its image under the matching. That is to say, when slice $z_0 = \text{round}(z)$ maps to slices $w_1, \ldots w_n$, transform vertices with slice coordinate z to the following slice coordinate w in the other volume:

$$w = n(z - \text{round}(z)) + \frac{1}{n} \sum_{i=1}^{n} w_i \tag{3}$$

An analogous approach can be taken for the latter case of slice compression.

3 Dynamic Programming

Dynamic Programming has been used in a variety of contexts for the purpose of sequence alignment. For instance, in computational biology, it is used to align DNA and protein sequences [4]. It has also been used in computer vision for object recognition and special cases of pose estimation. Chapter 5 of Aparna Ratan's Ph.D. Thesis [3] contains an survey of these applications, and presents a method for detecting a 2d template in an image, allowing for column-wise warpings of the template in a manner analogous to the technique presented here.

3.1 Optimal Substructure

Say we have two sets of volume slices, U_1^i indexed from 1 to i, and V_1^j indexed from 1 to j, that we would like to match in the manner described above. That is, we would like to allow for stretching and compression of slices under the matching, while preserving slice ordering. We can write down the optimal alignment recursively:

$$\binom{U_1^i}{V_1^j} = \binom{U_i}{V_j} \quad \circ \quad \text{best of} \begin{cases} \binom{U_1^i}{V_1^{j-1}} & , \text{ expansion of } U_i \\ \binom{U_1^{i-1}}{V_1^{j-1}} & , \text{ no expansion/compression} \\ \binom{U_1^{i-1}}{V_1^j} & , \text{ expansion of } V_j \end{cases} \tag{4}$$

where $\binom{U_1^i}{V_1^j}$ denotes the optimal alignment of subsequences U_1^i and V_1^m, and where $\circ$ denotes the concatenation of two subsequence alignments. Also note that $\binom{U_i}{V_j}$ simply means that these slices are aligned. Now, we pick a metric $F(U_1^i, V_1^j; M)$ to evaluate a matching M of the argument subsequences, such that it treats each matching of slices independently (say by summing their pair-wise similarities). Therefore can also write down the optimization of this metric recursively:

$$\max_M F(U_1^i, V_1^j; M) = f(U_i, V_j) \; + \; \max \begin{cases} \max_M F(U_1^i, V_1^{j-1}; M) & + \, g(\text{expansion of } U_i) \\ \max_M F(U_1^{i-1}, V_1^{j-1}; M) & + \, g(\text{no expansion/compression}) \\ \max_M F(U_1^{i-1}, V_1^j; M) & + \, g(\text{expansion of } V_j) \end{cases} \tag{5}$$

where $f(U_i, V_j)$ measures the value of matching slices U_i and V_j, i.e. some measure of their similarity. The function $g()$ is an additional term which may be used to disadvantage or encourage slice compression or expansion of different kinds. Since we would like slices to vary freely in this respect, we set $g() = 0$.

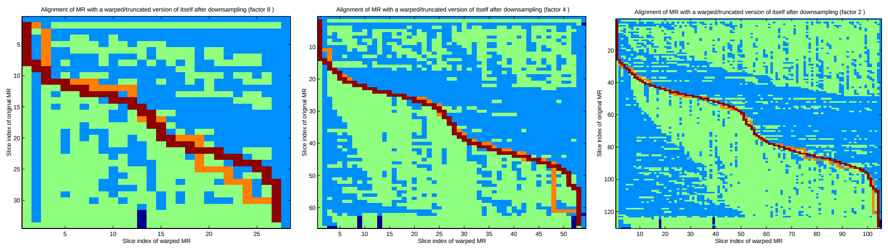
With this recursive expression for evaluating the value of an alignment, we can use Dynamic Programming in the usual manner to find the best matching between two sets of volume slices U_1^N and V_1^M. By the structure of the above recursion, we fill in an $N \times M$ matrix with entries at (i, j): $\max_M F(U_1^i, V_1^j; M)$ and pointer in the direction of the prefix sequence matching which produce this optimal value.

In practice, null slices may be appended to the beginning and end of both slice sequences, to account for the case where either or both volumes image structures beyond the range of the other. In this work, we set a constant value for matches to the null slice, which effectively serves as a threshold for starting to align real slices with one another.

3.2 Multiresolution Optimization

Even when the method for calculating the similarity of a matching of two slices is fast, filling the $N \times M$ Dynamic Programming matrix may take a lot of time. Since typical high resolution, anatomical MR or CT images may have hundreds of data slices, an $\Theta(NM)$ running time becomes restrictive. To reduce the bruden of this computation, we note that the solution to the our optimization does not rely on filling in every entry of the Dynamic Programming matrix. We do this only because we don't know where the solution path through this matrix (which is effectively the graph of the optimal matching) resides. If we had information about this, then we would only calculate those matrix entries near the solution path, setting the rest of the matrix entries to $-\infty$.

It happens that the alignment of downsampled version of volumes may produce good estimates of the alignment at higher resolution. This is illustrated in Figure 3 which shows the results of running our algorithm on a volume and a warped version of itself, both downsampled by factors of 2,4, and 8. The running time of the algorithm is much less for lower resolution volumes, not only because the total number of slices is less, but also because the slices themselves are smaller, so that it takes less time to evaluate their pairwise similarity. Therefore,



(a) factor 8 (b) factor 4 (c) factor 2

Fig. 3. At each level of isotropic downsampling, the maroon line plots the actual deformation between two MRI head volumes (approx. 256x256x128 voxels in original size) and the orange line plots the match calculated by dynamic programming. The other colors represent the "arrow" direction of each cell of the dynamic programming matrix. Slices of the original MRI are indexed along the vertical axis, and slices of the warped MRI, along the the horizontal (see Section 6)

we may use a multiresolution approach to solving this problem at high volume resolution. At the lowest level of image resolution, we calculate the full dynamic programming matrix. Then we use the derived matching to limit the region of interest (ROI) in the matrix at the next higher level of resolution. This is easy to do when levels of the pyramid differ by an integer factor, as illustrated in Figure 4. For each slice pairing (i, j) at low resolution, we add the corresponding slice pairs at higher resolution to the region of interest. Then, to account for the fact that the solution at low resolution may not be ideal, we dilate the region of interest in the dynamic programming matrix for the higher resolution alignment. Outside the region of interest, we set the matrix entries to $-\infty$, and run the dynamic programming optimization as usual.

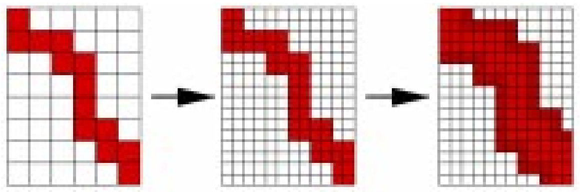


Fig. 4. The process of defining a region of interest for the dynamic programming matrix based on the solution of the lower resolution alignment.

We can continue this approach all the way up to the highest level of volume resolution, where the time savings will be most substantial. Since the number of slice pairs along a solution path is of order $\Theta(N + M)$, the region of interest will be of order $\Theta(k^2(N + M))$, much smaller than $\Theta(NM)$ for a large number of slices. Naturally, there is some risk in doing this. If the optimal solution falls outside the specified region of interest, then it will not be discovered. Nor is it clear, in general, how omitting an important part of the search space will purturb the rest of the solution. Therefore, we ought to be liberal in setting the dilation diameter k.

4 Intra-slice Registration

4.1 Similarity Measures

As described in Section 3.1, to use dynamic programming we must have some way of measuring the value/similarity of pairs of slices. Furthermore, since we would like our algorithm to run fast, we'd like to find some quick way of making this assessment.

For this purpose, we chose to use an intensity-based measure of similarity, such as correlation, local correlation, or mutual information. These measures are statistics on the joint intensity distribution of two images, for a given alignment of those images, and are each applicable in different contexts. For instance, correlation is an appropriate metric when we expect a linear (or at least a monotonic) relationship between voxel intensities, such as when we're registering volumes of the same imaging modality. When registering volumes of different modalities, such as CT and MR, we might choose Mutual Information which measures a statistical relationship between intensities, or Local Correlation[3] which is large when local intensity relationships are linear[2]. In accordance with the statistical interpretation of these similarity measures, we can estimate the their value on two images by means of a random sample of intensity pairs, rather than over all of them, to speed the calculation.

[3] That is, the average squared correlation coefficient over small patches of the image.

4.2 Alignment by Moments

Of course, before we can evaluate these measures, we must have registered the slices in some way. For this purpose, we propose using the centroid and the principle second moments of a rough binary segmentation of relevant structures. This approach has the advantage of being very fast relative to iterative approaches to this problem. In fact, the centroid and principle axes of a binary segmentation can also be estimated from a sample of the segmented points, rather than computing the complete sums.[4] Furthermore, these only need to be computed once per slice at the outset of registration, not every time a slice is compared to one of the other volume. For completeness, we point out that the approach is clearly a suboptimal method for image registration. In fact, it cannot fully capture a generic affine transform, which may have some degree of skew.

For each slice, we calculate the centroid and three 2^{nd} central moments of binary segmentation of relevant structures, normalizing by the area of the binary segmentation. Then we use eigendecomposition to get the principle axes and the area-normalized 2nd central moments in those directions. We denote the centroid as $\boldsymbol{\mu}$, the matrix of eigenvectors $\boldsymbol{V}$ (with columns ordered such that the determinant is positive) and the diagonal matrix of corresponding eigenmoments $\boldsymbol{D}$. Then, to register two slices, we find the affine transformation which

1. Maps one centroid to the other.
2. Rotates and scales one set of principle axes to match the other such that some measure of similarity is maximized.

Note that we do not simply align the major axes with each other, since this method might produce poor alignments in the case that the principle axes are of roughly the same size, or in the case that the true mapping from one slice to other maps the major axis to the minor, and vice versa. Furthermore, second moments alone do not indicate in which direction the moments should be aligned, so that blindly aligning two principle directions might yield an alignment which is 180 degrees misaligned.[5]

Instead, we compute an intensity-based measure of similarity for each of the four possible rotations of the principle axes onto the other, and pick the alignment which maximizes this value. Note that this is the similarity measure which is finally used in the Dynamic Programming framework. Specifically, for every permutation of the columns of the eigenvector matricies (adjusting signs so that the eigenvector matricies stay positive definite), we find the mapping $\boldsymbol{A}_{12}$ from one pair of eigenvectors (scaled by the root of their respective eigenmoments) to

[4] though we didn't do it in this work.

[5] One possible way to disambiguate this alignment that is in keeping with the approach of alignment by moments, would be to calculate the third central moment in each of the principle directions, and then align axes based on skew. However, this is probably less robust and less computationally attractive than using gray level information.

the other

$$\boldsymbol{D}_2^{\frac{1}{2}}\boldsymbol{V_2} = \boldsymbol{A}_{12}\boldsymbol{D}_1^{\frac{1}{2}}\boldsymbol{V_1} \tag{6}$$

$$\boldsymbol{A}_{12} = \boldsymbol{D}_2^{\frac{1}{2}}\boldsymbol{V_2}\boldsymbol{V_1}^T\boldsymbol{D}_1^{-\frac{1}{2}} \tag{7}$$

and choose the resulting alignment which maximizes some similarity measure. For each slice pairing, we save the intraslice transformation for use in creating the complete deformation field after the Dynamic Programming step. This positive definite, affine transform is

$$\boldsymbol{x}_2 - \boldsymbol{\mu}_2 = A_{12}(\boldsymbol{x}_1 - \boldsymbol{\mu}_1) \tag{8}$$

$$\boldsymbol{x}_2 = A_{12}\boldsymbol{x}_1 + \underbrace{\boldsymbol{\mu}_2 - A_{12}\boldsymbol{\mu}_1}_{\boldsymbol{t}_{12}} \tag{9}$$

$$\tag{10}$$

4.3 Application to MRI Head Images

We will illustrate this approach by registering two corresponding slices of MRI head volumes. The code which produced these results can be found in moments.m, momentreg.m, momentcorr2.m, momentdisp.m, segMRhead.m, and imtrans.m. The first step in the process is to segment the head. This is relatively easy to do since the head is surrounded by air, which has uniformly low intensity in MRI images. We can segment the air by intensity thresholding; however, this does not properly segment the head[6] due to the fact that anhydrous regions inside the head (such as bone and air in the nasal cavities) also have low intensity in MRI, as shown in Figure 5. We can distinguish the air around

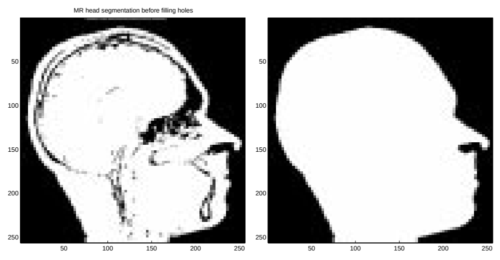


Fig. 5. Head segmentation by thresholding followed by morphological operations.

the head by performing connected components analysis on the initial air segmentation, and picking the largest connected component as the air surrounding the head. Alternatively, we could use grassfire or region growing methods with

[6] Though, in truth, it may be sufficient to properly calculate moments.

seed points outside the head.[7] Before we can do this, however, we need to block connections between air inside and outside the head. To do this, we run a slice-wise morphological closing operation on the initial head segmentation to seal the ear canals and nose. Finally, we clean up some isolated "head" voxels outside the head with an opening operation. These steps are coded in the script segMRhead.m.

Once the segmentation is complete, we calculate the moments of the two head slices, shown superimposed over the images in Figure 6 (a,b). The result of the alignment by moments is shown in Figure 6 (c). Note that we used correlation coefficient to evaluate the four possible alignments.

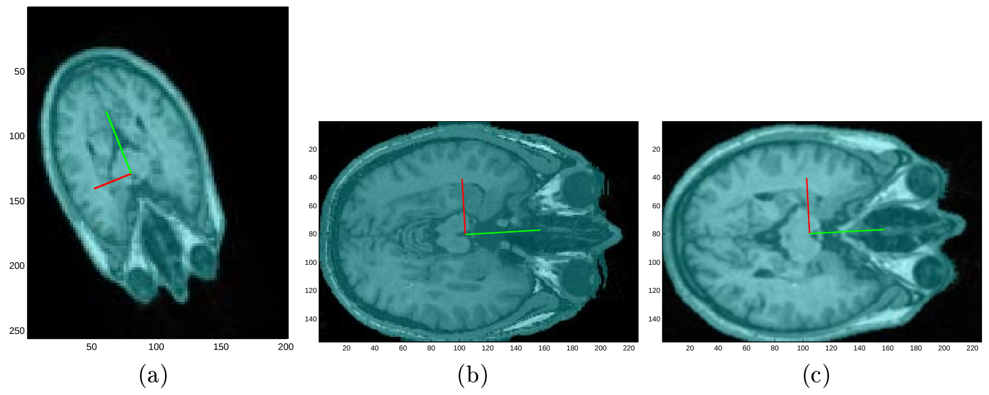


Fig. 6. Registration by moments of corresponding T1-weighted head slices of different subjects, taken in different scanners. The transformation of image (a) into the coordinates of image (b) is shown in image (c). The cyan tint shows the binary head segmentation with which the superimposed principle axes were calculated.

5 Algorithm

The following ourlines the steps of our algorithm. The details of each step can be found in previous sections.

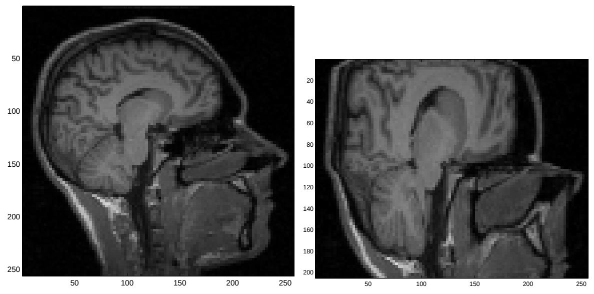
1. Segment the structures to be aligned in each volume.
2. Select a slice direction for each volume, resampling to align it with a data axis.
3. Calculate the centroid and principle axes of each slice of each volume.
4. Create a multiresolution pyramid of images, downsampling along each data axis by a factor of 2 at each level.

[7] Actually, we used a built-in MATLAB function for hole filling because it was faster.

5. For each level of downsampling, starting with the lowest resolution and considering all slice pairs:
   (a) For each slice pair (i, j) under consideration:
      i. Align by centroid and principle axes.
      ii. Evaluate Similarity (e.g. by sample correlation)
   (b) Find the optimal slice sequence matching by Dynamic Programming.
   (c) Repeat for the next higher resolution level, considering only those slice pairs (i, j) consistent with the matching found at the current resolution.

6 Experiments

We tested our method on partially synthetic data so that we would have some ground truth to reference to the registration results. We took a T1-weighted MRI head volume of size 256x256x128 and warped its slices normal to its second index direction (axial slices) into a volume of size 256x205x128. A slice normal to the third index direction are shown for each volume in Figure 7. The warp was limited to expansion and compression in a single direction, and the volume was truncated so that not all structures appeared in the warped volume. As is clear from the figure, the warp was a rather large one. To simplify our analysis of the results, we chose not to introduce any in-slice distortion or any misalignment of the slice directions.



(a) (b)

Fig. 7.

For our first experiments, we did not use a multiresolution approach for these experiments, but rather calculated the full dynamic programming matrix for isotropic downsampling factors of 2, 4, and 8. The actual warp and the calculated warp are shown superimposed on the dynamic programming matrix for each level of downsampling in Figure 3. The calculated warps at each level are not inconsistent with the actual warp, and the calculated warp at downsample factor 2 is very good. Especially at the lower levels of resolution, the algorithm had trouble at the ends of the alignment, where the warped volume had no slices corresponding to those in the original volume. These locations also correspond to regions at the top of the head and neck where structures have similar appear from axial slice to slice (except for scaling). The running times in MATLAB on a Pentium III, 500 MHz, dual processor system, are shown for each level of downsampling in the table below. Note that the running time for an alignment at full resolution without the use a our hierarchical approach was too long to fully test. We would expect these run times to improve substantially were the code complied and not just interpreted by MATLAB.

Factor	**Original Size**	**Warped Size**	**Run Time**
8	32 x 32 x 24	32 x 26 x 24	18 sec
4	64 x 64 x 48	64 x 52 x 48	2.5 min
2	128 x 128 x 96	128 x 103 x 96	30 min

Next we applied the multiresolution approach to the same data, arbitrarily choosing the factor by which lower resolution solutions are dilated to determine the region of interest (ROI) in the Dynamic Programming Matrix at higher resolution. We used a pyramid fo downsample factors (8,4,2,2,1), reducing the dilation factor toward the top. This was motivated by our previous observation that the level 2 registration produced good results. The results are displayed in Figure 6. It's clear that bad behavior at lower resolutions (in the lower right hand corner of Figure 6 (a,b)) created localized problems for the alignment at higher resolutions. This particular situation suggests that there may be more intelligent ways to dilate low-resolution solutions, for instance by dilating asymetrically, locally biasing toward the direction that the solution curve moved in the previous two levels of the hierarchy. More importantly, it seems that using the full resolution image doesn't improve the solution very much. It does, however, add substantially to the overall running time of the algorithm. In this case, we might have been better off just computing the level 2 solution, starting with a broader ROI.

Factor	size ROI	Dilation Diam.	Run Time
8	32 x 26	N/A	18 sec
4	1552	15	1.3 min
2	3269	15	8.5 min
2	1459	7	3.8 min
1	3303	5	14.5 min

7 Potential Applications

In this section, we'll discuss a few specific application domains in more detail than we have up to this point.

The slicewise registration has the potential for application in intrasubject image registration due to the fact that the same physical structures are present, and are possibly present in corresponding slices of the data. The distortion that we're trying to recover may be due to actually physical distortion, or to distortions due to the imaging modality or the particulars of the imaging apparatus.

An example of the latter type of distortion can be encountered in registering fMRI and anatomical MRI of the same head. Due to large slice thickness and the fast changing fields involved in functional MRI (EPI) acquisitions, nonlinear distortions may appear in fMRI images that are not present in high-resolution anatomical images. One example of this is a signal drop and apparent slice compression in areas around the nasal cavity, and other regions of susceptibility variabiliy through the slice. Rigid registration techniques, such as those by Mutual Information, often fail as a result. Since the data axes of head images often align well with "anatomical axes" due to the way subjects are positioned in scanners, a slicewise registration approach might be applicable here.

Such as system might also be useful in registering images of elastic organs in which compression occurs primarily in one direction. Because such images would be of the same physical structure, it is likely data slices orthogonal to this direction would contain roughly the same structures warped by some affine transform (probably also primarily scaling). Therefore, if such data volumes were resampled so that one data axis is aligned with this direction, and we might use this dynamic programming approach to estimate the deformation.

For instance, in the case of prostate imaging for the diagnosis and localization of tumors, it is often useful to align a 3d image taken with a typical MR body scanner, and those taken using a transrectal RF coil. Registering such images is

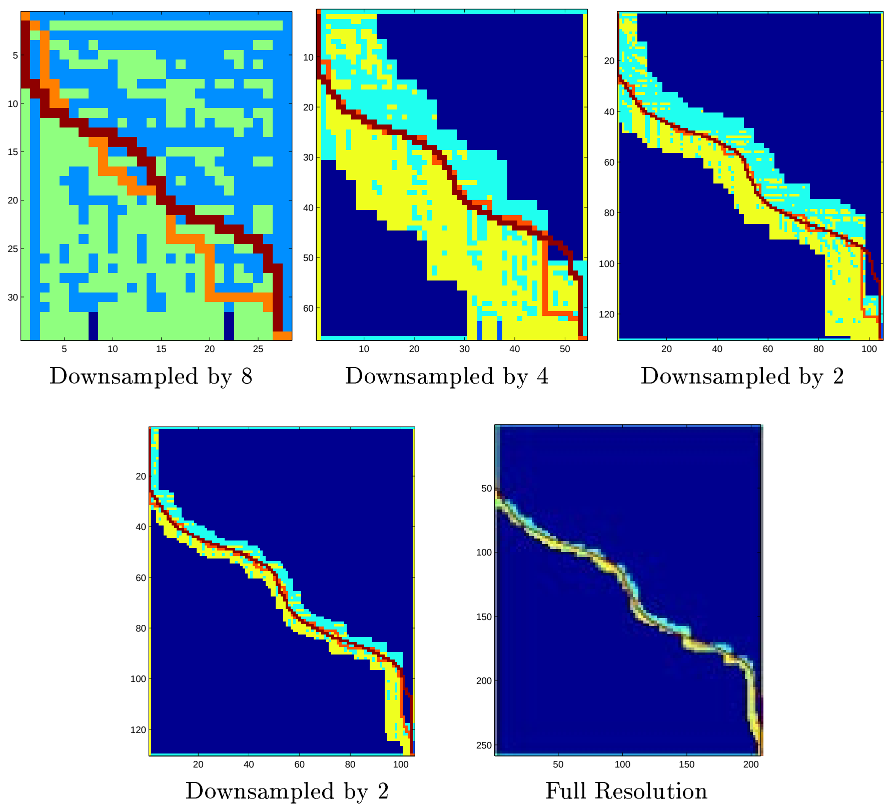


Fig. 8. A multiresolution registration. The maroon curve plots the actual deformation in slice thickness, whereas the orange or red curves plot the computed deformation at each level of downsampling. Dark blue denote the regions of the dynamic programming matrix whose values were set to $-\infty$.

an area of current research because the coil placement causes a drastic compression of the prostate, which can frustrate current non-rigid alignment techniques. If the compression is roughly unidirectional and nonuniform, then this approach might find an application.

Recovering deformations of the liver are another possible application area. Liver deformation is very large due to movement of the diaphragm during breathing. The primarily unidirectional motion of the diapragm suggests the possible utility of this approach.

References

1. T. H. Cormen, C. E. Leisterson, and R. L. Rivest. *Introduction to Algorithms.* McGraw-Hill, New York, 1998.
2. T. Netsch, P. Rosch, A. van Muiswinkel, and J. Weese. Toward real-time multimodal 3-d medical image registration. *ICCV*, 2001.
3. A. L. Ratan. Learning visual concepts ofr image classification. *Ph.D. Thesis, A.I. Lab, MIT*, 1999.
4. J. Setubal and J. Meidanis. *Introduction to Computational Molecular Biology.* PWS Publishing Co., New York, 1997.