Using Region Trajectories to Construct an Accurate and Efficient Polyaffine Transform Model

Gang Song, Yang Liu, Baohua Wu, Brian Avants, and James C. Gee

Penn Image Computing and Science Lab,
University of Pennsylvania, Philadelphia, PA, USA
songgang@seas,liuyang@sas,baohua@seas,
avants@grasp,gee@mail.med.upenn.edu
http://picsl.upenn.edu

Abstract. In this paper we propose a novel way to construct a diffeomorphic polyaffine model. Each affine transform is defined on a local region and the resulting diffeomorphism encapsulates all the local transforms by a smooth and invertible displacement field. Compared with traditional weighting schemes used in combining local transforms, our new scheme guarantees that the resulting transform precisely preserves the value of each local affine transform. By introducing the trajectory of local regions instead of using regions themselves, the new approach encodes precisely each local affine transform using a diffeomorphism with one or more stationary velocity fields. Experiments show that our new polyaffine model is both accurate and efficient.

Keywords: Polyaffine, Transform, Diffeomorphism.

1 Introduction

The transform models applied in image registration have a wide span of degrees of freedom. The transform can be as simple as an affine transform [1], which is a linear function defined on the whole image domain and only requires 12 scalar parameters for an image of three dimensions. In contrast, one can also use a deformation field as the transform in a non-rigid image registration [2, 3], which has an arbitrary number of degrees of freedom at the cost of expensive computation and difficult optimization. Between these extremes of parameterization, many other transforms have been studied, such as B-Splines used in free-form deformation [4], Geodesic Interpolating Splines [5] and finite element method [6]. These transforms are capable of describing a wide range of non-rigid transforms while using fewer parameters than a dense displacement field.

The polyaffine transform is a parameterization for deformable maps that fills the gap between a global affine transform and a deformation field transform. It exploits the prior knowledge that for many applications of medical image registration, the underlying anatomical structure is comprised of multiple local

J.C. Gee et al. (Eds.): IPMI 2013, LNCS 7917, pp. 668–679, 2013.

regions. An example of two lobes moving in different directions is illustrated in Fig. 1. Each local region is roughly rigid and can be approximated by a different local affine transform. The Polyaffine transform is a mathematical framework that fuses the local affine transforms through one non-rigid transform. Among different approaches like the piecewise affine transform [7, 8], a Log-Euclidean framework was first proposed in [9] especially to construct a smooth and invertible diffeomorphism, which can also be computed efficiently. This framework has been adopted in multiple image registration applications [9–13].

The key concept in [9] is to construct a stationary velocity field by fusing multiple regions with different affine velocity. The stationary velocity field is further integrated over time to generate an invertible transform. The construction method used in [9], however, as discussed in Section 2, cannot guarantee that the resulting final transform gives the exact same value of the input transforms in each local region 2. Our work reformulates the polyaffine transform as finding a feasible solution to a constrained problem. This new approach guarantees the local affine transformations are preserved. To achieve this, we demonstrate that the weight function used to fuse affine velocities has to be defined using a time-varying function in the framework of diffeomorphisms. The trajectory of each local affine region is proposed for computing the weight function in modeling a time-varying diffeomorphism with a series of stationary diffeomorphic transforms. With this new concept of region trajectory, our approach preserves each local transform and remains efficient in its implementation.

2 Methods

Consider a set of K affine transforms, $T_i(x) = A_i x + b_i$, each defined on a local region M_i. We want to integrate these local transforms into one transform $\phi(x)$ defined on the whole image domain Ω. For simplicity, these K local regions are subregions of Ω and do not overlap with each other. The polyaffine problem can be formulated as finding a feasible solution $\phi(x) : \Omega \to \Omega$ satisfying the constraints

$$\phi(x) = A_i x + b_i \text{ , when } x \in M_i \tag{1}$$

A direct way to construct such a ϕ is by simply averaging each local affine transform using a weight function w_i as in [14]: $\phi(x) = \sum_{i=1}^{K} w_i(x) T_i(x)$. A popular choice of weighting function is defined by the distance from the point x to each mask M_i: $w_i(x) \propto \exp(-\frac{\text{dist}(x,M_i)}{\sigma^2})$. Then the K weights $w_i(x)$ computed on the location x are further normalized such that $\sum_i w_i(x) = 1$.

This transform is smooth in the sense of its deformation gradient. However, one significant drawback is that it is not invertible in general [9]. To obtain the invertibility, i.e. making ϕ a diffeomorphism, the velocity field $v(x,t)$ was introduced in constructing ϕ in [9].

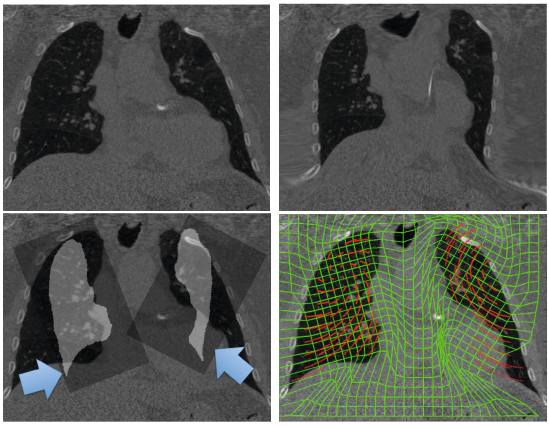


Fig. 1. Computed tomography images of two lung lobes moving in different directions. Top row: images before/after applying the transform. Bottom row: the two local affine transforms and the computed polyaffine transform using the proposed approach.

2.1 Diffeomorphism in Polyaffine Model

A diffeomorphism ϕ can be obtained by solving the ordinary differential equation over a time variable t:

$$\frac{\mathrm{d}\phi(x,t)}{\mathrm{d}t} = v(\phi(x,t),t) \tag{2}$$

At time $t = 1$, the diffeomorphism $\phi(x,1)$ can be obtained by integrating the velocity field $v(x,t)$ over time from $t = 0$ to 1. The affine transform $T(x) = Ax+b$ is an example of such a diffeomorphism. Using homogenous coordinates, an affine transform can be generated by a velocity field v [9]:

$$v(x,t) = Lx + u \text{ , with } \begin{bmatrix} L & u \\ 0 & 0 \end{bmatrix} = \log\left(\begin{bmatrix} A & b \\ 0 & 1 \end{bmatrix}\right) \tag{3}$$

Arsigny et al. proposed in [9] to first construct a velocity field v by fusing the K local affine velocity fields:

$$v(x) = \sum_i w_i(x)v_i(x), \text{ with } v_i(x) = L_i x + u_i, w_i(x) \propto \exp\left(-\frac{\mathrm{dist}(x, M_i)}{\sigma^2}\right) \tag{4}$$

It should be noted that here $v(x)$ is independent of t, i.e. $v(x) = v(x,t)$. Thus it is a stationary velocity field. To integrate ϕ from the velocity field v, a general approach is to discretize time from $t = 0$ to 1 into N time points. In the special case when v is stationary, an efficient recursive scaling-and-squaring method was proposed in [9] to compute the total field in only $\log N$ steps.

2.2 Polyaffine Transform Preserving Trajectories

Introducing the velocity field v as the generator for the transform guarantees the invertibility of the final transform. This, however, adds a new constraint in

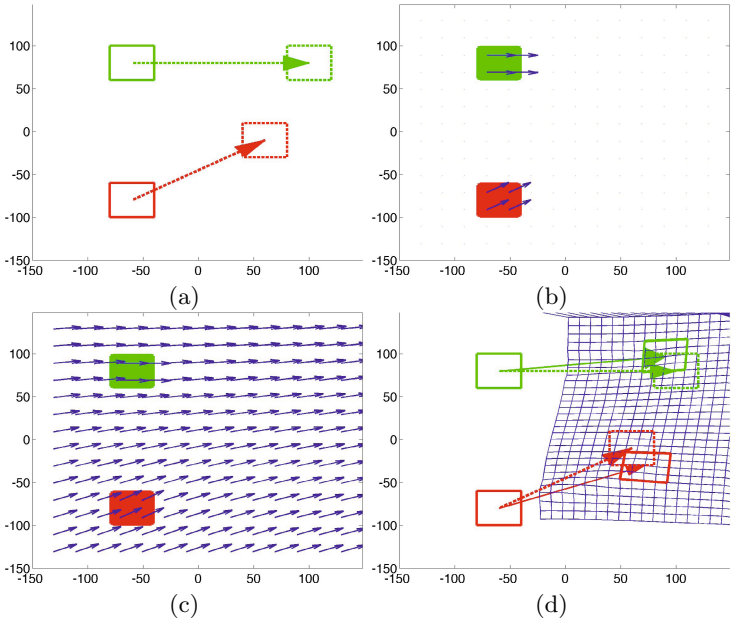


Fig. 2. Polyaffine transform using the previous approach [9]. (a) Two input affine transforms are shown in red and green. The starting and ending locations are indicated by the boxes in solid and dashed lines respectively. (b) The input local regions (shown as red and green areas) for each transform with their velocity plotted inside. (c) Computed velocity field using Eqn. 4, interpolated from the input local regions. (d) The resulting transform. The ending locations of each local region are plotted as boxes in solid lines at the arrow heads. The solid arrows (the result) deviate from the dashed arrows (the input), showing that local transforms are not preserved precisely.

finding a feasible solution to Eqn. 1. Instead of requiring the resulting transform $\phi(x)$ to match each local affine transform $T_i(x)$ within M_i at $t = 1$, it requires the velocity $v(x)$ to match each transform at all times from $t = 0$ to 1. Thus we can specialize the problem of Eqn. 1 into a more restricted one:

$$v(\phi(x,t),t) = v_i(\phi(x,t),t) \text{ for } x \in M_i \tag{5}$$

The solution of the new problem Eqn. 5 ensures that the whole temporal trajectory (from $t = 0$ to 1) for all x in M_i matches T_i. Indeed, given the uniqueness of the solution to the O.D.E of Eqn. 2, we have $\phi(x,t) = T_i(x,t)$ for $x \in M_i$ since $T_i(x) = A_i x + b_i$ is the solution when$v(\phi(x,t),t) = v_i(\phi(x,t),t)$. A natural solution of $\phi(x,t)$ in Eqn. 5 is to use a time-varying weight function to define the time-varying velocity field at any time t:

$$\begin{aligned} v(y,t) &= \sum w_i(y,t) v_i(y) \text{ , with } y = \phi(x,t) \text{ and} \\ w_i(y,t) &\propto \exp\left(-\frac{\text{dist}(y, \phi(M_i,t))}{\sigma^2}\right) \end{aligned} \tag{6}$$

The weight $w_i(y,t)$ needs to be defined over t by tracking each M_i at time t, $\phi(M_i,t) = \{\phi(x,t), x \in M_i\}$. If a point $y = \phi(x,t)$ belongs to the i-th transform, the weight w_i should have:

$$w_i(y,t) = 1 \text{ and } w_{j\neq i}(y,t) = 0 \text{ for } x \in M_i \ . \tag{7}$$

This new definition of velocity field is different from the one used in [9], i.e. Eqn. 4, where it only matches v_i at time $t = 0$. The new definition of the weight w_i depends on t and needs to track the trajectory of the local mask M_i.

The definition in Eqn. 4 [9] could not guarantee that the velocity $v(\phi(x,t),t)$ is still dominated by T_i when a point $x \in M_i$ at $t = 0$ moves to a new location $\phi(x,t)$ at time t. Thus it could not preserve the trajectory of each local affine region. This makes it an inviable solution to Eqn. 5 (illustrated in Fig. 2).

2.3 Extend Local Region to Trajectory of Local Region

Although Eqn. 6 gives a feasible solution to Eqn. 5, it also eliminates the nice property of stationary velocity and is inefficient in computation. We propose a novel way to define a stationary velocity field v which still satisfies Eqn. 5. Define the trajectory of the region M_i from time t_1 to t_2 as notation $M_i^*|_{t_1}^{t_2} = \cup_{\tau=t_1}^{t_2} \phi(M_i,\tau)$.

Without loss of generality, we first assume that these region trajectories do not overlap in the spatial domain Ω. Define a new stationary weighting function $w(x)$ independent of t:

$$\begin{gathered} v(y) = \sum w_i(y) v_i(y) \\ \text{with } w_i(y) \propto \text{dist}(y, M_i^*|_0^1) \end{gathered} \tag{8}$$

Since for $i \neq j$, $M_i^*|_0^1 \cap M_j^*|_0^1 = \varnothing$, for $x \in M_i$ we have $\text{dist}(\phi(x,t), M_i^*|_0^1) = 0$ and $\text{dist}(\phi(x,t), M_{j\neq i}^*|_0^1) \gg 0$. Thus the new stationary $w(x)$ still satisfies the same property of the time-varying version of $w_i(x,t)$ in Eqn. 7: $w_i(T_i(x,t)) = 1$ and $w_{i\neq j}(T_i(x,t)) = 0$ for $x \in M_i$. By introducing the region trajectory $M_i^*|_0^1$, the proposed stationary weight w_i gives the same ϕ as the time-varying version in Eqn. 6. The only difference between the two resulting ϕ is in the intermediate areas outside of $\cup_{i=1}^K M_i$.

When comparing the proposed stationary weight in Eqn. 8 and its generalized time-varying version in Eqn. 6, one should notice that $w(y,t)$ in Eqn. 6 is defined on the spatial-temporal domain $\Omega \times [0,1]$, while $w(y)$ in Eqn. 8 is a "squeezed version" that collapses the region trajectory M_i^* along the temporal axis into the spatial domain. The result using the proposed weight for the case in Fig. 2 is shown in Fig. 3, where our solution clearly preserves the input affine transform.

2.4 Series of Stationary Velocity Field for Trajectory Collision

One critical assumption in eliminating t from Eqn. 6 is that all trajectories $M_i^*|_0^1$ do not overlap in the spatial domain, $M_i^*|_0^1 \cap M_j^*|_0^1 = \varnothing$. Otherwise it will be

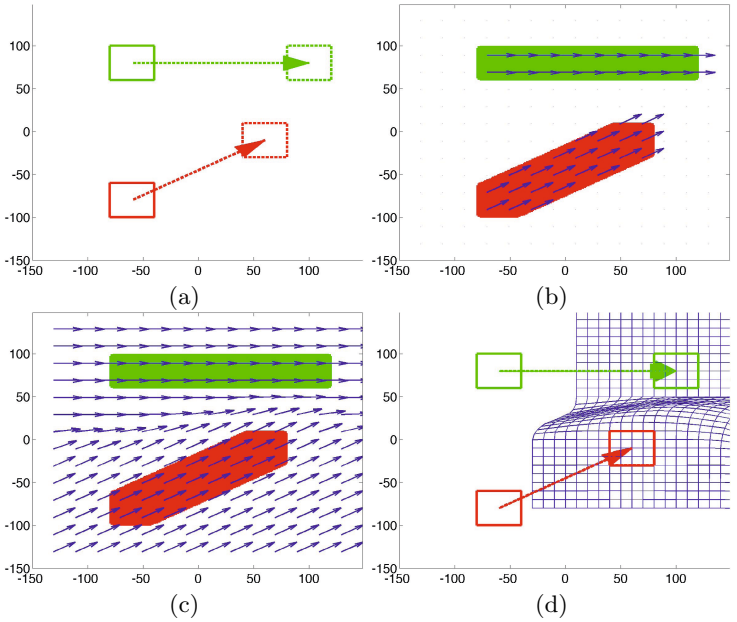


Fig. 3. Polyaffine transform using our proposed method. (a) Two input affine transforms (same as in Fig.2(a)) are shown in red and green. The starting and ending locations are indicated by the boxes in solid and dashed lines respectively. (b) The trajectories of the input local regions (shown as red and green areas) for each transform with their velocity plotted inside. (c) Computed velocity field using Eqn. 8, interpolated from the input region trajectories. (d) The resulting transform. The ending locations of each local region are plotted as boxes in solid lines at the arrow heads. The solid arrows (the result) are overlapped with the dashed arrows (the input), showing that local transforms are preserved precisely.

ambiguous to determine which local affine transform should be used when a point y belongs to multiple trajectories.

This non-collision assumption is nevertheless not true in general. Consider two local affine transforms T_1 and T_2 in Fig. 4, defined on region M_1 and M_2 respectively. The end of trajectory of M_1, $T_1(M_1)$, is overlapped with the starting position of M_2. In this case, for point y inside $T_1(M_1) \cup M_2$, it is ambiguous to define its weight $w_i(y)$ using Eqn. 8.

Our solution to this dilemma is to find a period from time t_1 to t_2 so that the trajectories within this period are not overlapped. By the time $t = 1$ when M_1 moves to $T_1(M_1)$, M_2 also moves to $T_2(M_2)$ and $T_1(M_1) \cap T_2(M_2) = \varnothing$. In general we need at any time $\tau_1, \tau_2 \in [t_1, t_2]$, no local regions are overlapped, $\phi(M_i, \tau_1) \cap \phi(M_j, \tau_2) = \varnothing$. When these trajectories are disjoint in the spatial-temporal domain, such $[t_1, t_2]$ is feasible. Thus it is possible to break the time from 0 to 1 into a sequence of $C + 1$ time points $[t_0, \ldots, t_C]$, such that

$$t_0 = 0 \text{ and } t_{k+1} = \max_{\tau} \{\tau | M_i^* |_{t_k}^{\tau} \cap M_j^* |_{t_k}^{\tau} = \varnothing, \forall i \neq j\} \tag{9}$$

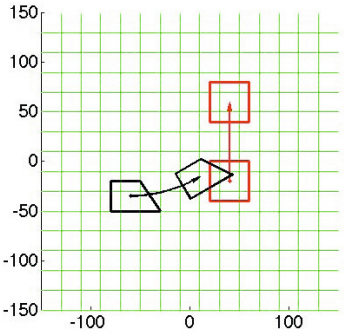


Fig. 4. The trajectories of two local regions overlap. Each arrow represents an input local transform, plotted in black and red. The boxes at the arrow tails are the starting locations of each input transform; the boxes at the heads are the ending locations.

For each non-collision period $[t_{k-1}, t_k]$, a stationary velocity v^k and its associated weight function are defined in the same way as in Eqn. 8. For each stationary velocity v^k, its transform ϕ^k is computed using the efficient scaling-and-squaring method in [9]. The final transform ϕ is the concatenation of these C diffeomorphism transforms and is also a diffeomorphism: $\phi = \phi^C \circ \cdots \circ \phi^2 \circ \phi^1$.

3 Implementation

We have proposed the notion of region trajectory to construct the polyaffine model. Unlike previous models, our polyaffine transform guarantees that the final transform is the exact value of $T_i(x)$ in each individual local region M_i. Before we discuss some of our implementation details, here is the summary of the construction steps:

1. Compute the collision time points $\{t_0 = 0, t_1, \ldots, t_C = 1\}$ using Eqn. 9.
2. Construct the stationary velocity v^k for each collision period $t = t_{k-1}$ to t_k.
3. Compute the corresponding diffeomorphism ϕ^k from v^k.
4. Concatenate all ϕ^k to get the final diffeomorphism ϕ.

3.1 Choice of Weight Function

A common practice to define the weight function $w_i(x)$ is to use the distance between the point x to the region M_i [9], or its trajectory M_i^* in our case. The weight is computed using a Gaussian function [9], or other decreasing functions [10]. The weights are further normalized by scale so that $\sum_i w_i(x) = 1$.

There are two drawbacks of using directly $\text{dist}(x, M_i^*)$. The first is a practical issue. For x in M_i^*, its distance to the j-th transform region trajectory cannot be infinite, and $w_{j \neq i}(x)$ is in general a small, nonzero scalar after

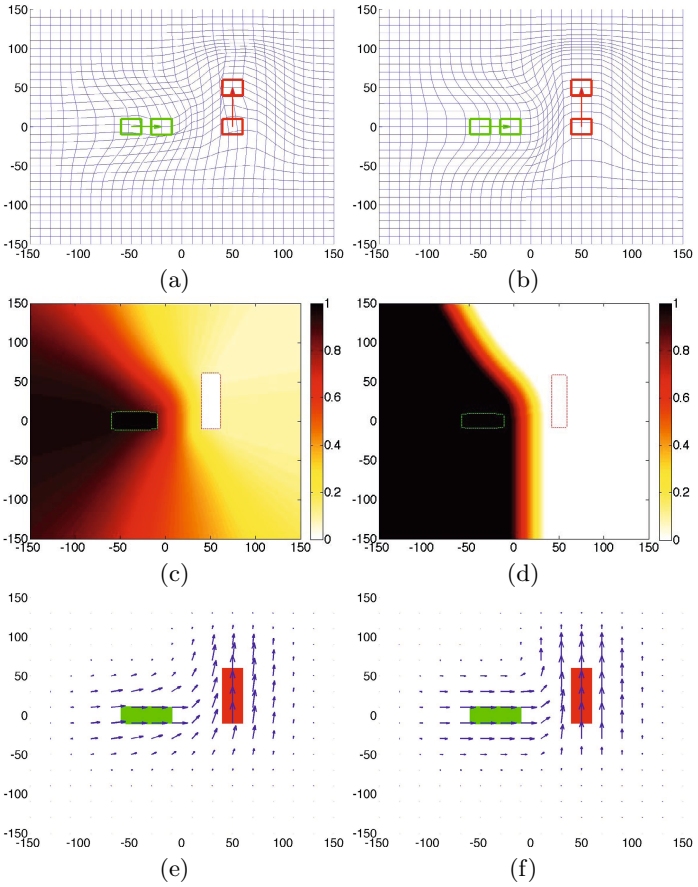


Fig. 5. Comparison of the transforms computed using (a,c,e) $\mathrm{dist}(x, M_i^*)$, and (b,d,f) $\mathrm{dist}(x, \partial\overline{M_i^*})$. (a,b) The transforms are constructed from the same input, one horizontal translation (green arrow) and one vertical translation (red arrow). (b) is visually more smooth than (a). (c,d) The weights $w(x)$ for the local vertical translation, range in $[0, 1]$. (d) is more polarized except along boundary $\partial\overline{M_i^*}$. (e,f) The resulting velocity below the green trajectory has an upward vertical component in (e) but not in (f). Similar effects are seen on the right of the red trajectory. M_i^* are shown as green and red dotted boundaries in (c,d) and solid areas in (e,f).

normalization. Some arbitrary thresholding needs to be performed to approximate $w_{j\neq i}(x) = 0$. Finding a suitable σ in the Gaussian function for different regions is also arbitrary.

The second drawback is that such a distance does not consider the relative spatial configuration of different region trajectories. For example, in Fig. 5(b), there are two local transforms. One is a horizontal translation (on left), and the other is a vertical translation (on right). By examining the vertical component of the velocity field below the horizontal trajectory, it can be seen that there is

some upward vertical velocity component, which comes from the small weights on the right vertical translation. Ideally, there should be no such vertical velocity around this region since the vertical velocity should be *blocked* by the horizontal trajectory on its left.

To eliminate this drawback, we propose a new weighting scheme by dividing the image into subdomains. The subdomains $\{\overline{M_i^*}\}$ are a partition of the whole image domain Ω. The subdomain $\overline{M_i^*}$ can be viewed as the region of influence of the trajectory M_i^* in the center. Based on this intuition, we use the distance to each trajectory to define the subdomain $\overline{M_i^*}$ as the the area containing all the points closer to M_i^* than other trajectories:

$$\overline{M_i^*} = \{x \in \Omega | \text{dist}(x, M_i^*) = \min_{j=1\ldots K} \text{dist}(x, M_j^*)\} \tag{10}$$

Such a definition partitions the whole image domain Ω into K disjoint subdomains. The proposed weight is correspondingly defined by the distance from the point x to the boundary of the subdomain $\overline{M_i^*}$. If x is close to the boundary, the weights should be nonzero for all adjacent local transforms. If x is in the center and far away from the boundary, its weight should be polarized. Note that $\text{d}(x) = \text{dist}(x, \partial\overline{M_i^*})$, and σ is a constant. We can define w_i before normalization using a simple piecewise linear function.

$$w_i(x) \propto \begin{cases} 1 & \text{if } x \in \overline{M_i^*} \text{ and } \text{d}(x) > \sigma \\ 0.5 + \text{d}(x)/2\sigma & \text{if } x \in \overline{M_i^*} \text{ and } 0 \leq \text{d}(x) < \sigma \\ 0.5 - \text{d}(x)/2\sigma & \text{if } x \notin \overline{M_i^*} \text{ and } 0 \leq \text{d}(x) < \sigma \\ 0 & \text{if } x \notin \overline{M_i^*} \text{ and } \text{d}(x) > \sigma \end{cases} \tag{11}$$

An example of the computed weight function and the resulting transforms on two local affine transforms is illustrated in Fig. 5. Such a weighting scheme takes consideration of the relative position of each trajectory mask. Along the boundaries of the subdomains are the locations where the weighting matters most, while inside each subdomain it is dominated by one local transform. This can be viewed as an efficient approximate solution to the following equation:

$$w_i^* = \min \int \|\nabla w_i\|^2 dx, \text{where } w_i(x) = 1 \text{ for } x \in M_i \text{ and } w_i(x) = 0 \text{ for } x \in M_{j \neq i} \tag{12}$$

In one extreme case when each local region is one point and uniformly distributed, such a partition becomes the Voronoi diagram on the image domain if time $t = 0$ is only considered.

3.2 Computing Mask Trajectories and Their Collision

Another important implementation detail is how to efficiently compute each region trajectory M_i^* given its predefined velocity $v_i(x) = L_i x + u_i$. This is trivial when M_i is just a single point. However, when M_i is a region, there could be multiple trajectories. When a region can be efficiently parameterized

as a polyhedron, one can track the trajectory of each vertex and compute the collision of any two polyhedra. Here we instead choose a simpler approximation by using a random point set to represent an arbitrary region.

For a region M, a point set $\{p_a\}$ is uniformly sampled inside using a sampling diameter d. The region can be then approximated by the union of dilations from the point set $\{p_a\}$ with distance d. The region M transformed at time t, $T(M,t)$, is also approximated by the dilation of the point set $\{T(p_a,t)\}$. The condition of collision detection in Eqn. 9 is implemented using:

$$M_i^*|_{t_1}^{t_2} \cap M_j^*|_{t_1}^{t_2} = \varnothing \Leftrightarrow \min_{\tau_i=t_1}^{t_2} \min_{\tau_j=t_1}^{t_2} \min_{a,b} \operatorname{dist}(T_i(p_{i,a},\tau_i), T_j(p_{j,b},\tau_j)) < d \qquad (13)$$

One interesting fact about the distance between the two point trajectories of a local affine region is that it is well bounded by a convex function of time variable t when the region deforms over time. For simplicity without using extra notations in the homogenous coordinates, suppose L is the logarithm of the affine matrix A (see Eqn. 3), we have $T(x,t) = \exp(tL)x$. We have the following theorem.

Theorem 1. *$f(t) = \|\exp(tL)(p-q)\|$ is bounded by a convex function for a given matrix L and any two points p and q.*

Proof 1

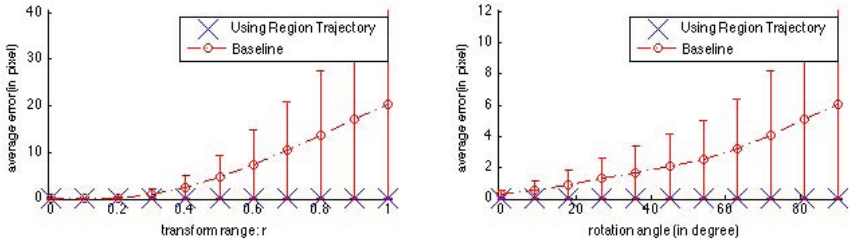
$$\|\exp(tL)(p-q)\| \leq \|\sum_{n=0} \frac{(tL)^n}{n!}\|\|p-q\| \leq \sum_{n=0} \frac{(t\|L\|)^n}{n!}\|p-q\| = \exp(t\|L\|)\|p-q\|$$

This bound ensures that if a sampling diameter d is small enough, the dilation of the point set $\{T(p,t)\}$ at time t should cover the mask $T(M,t)$ and thus is a good approximation.

4 Results

We evaluated the accuracy of our proposed approach using synthetic experiments. Two translation transforms were defined on two rectangular regions, similar to the inputs in Fig. 5. A scalar $r \in [0,1]$ was used to control the relative range of the translation offset, $t_1 = r \times [100,0]$ and $t_2 = r \times [0,50]$. When $r = 0$, they were identity transforms; when r increased, the two affine trajectories overlap. The accuracy was evaluated by the difference between the resulting transform and the corresponding affine transform for each pixel in all regions, $e(x) = \|\phi(x) - (A_i x + t_i)\|$. The mean and variance of all $e(x)$ for $x \in M_1 \cup M_2$ were computed versus r.

We compared our proposed polyaffine construction using the region trajectory with the approach in [9] as the baseline, where the weight was computed based on the region only. When the translation was relatively small ($r < 0.3$), both methods had a good accuracy. However, when the translation increased, the error in the baseline method also increased quickly since the region trajectories became significantly different from the regions. In contrast, our approach still maintained a high accuracy with near-zero error. Similar results could be observed when comparing the case of two rotation transforms by changing the rotation angles (Fig. 4(b)).



(a) (b)

Fig. 6. Comparison of our proposed method and the baseline method [9], using (a) two local translation transforms and (b) two local rotation transforms. The x-axis corresponds to (a) the translation offset, or (b) the rotation angle. The y-axis corresponds to the average error of the resulting transforms in the local affine regions, with standard deviations plotted as error bars.

5 Discussion

In this work we presented a novel approach to constructing a polyaffine transform which can precisely preserve each affine transform using one diffeomorphism. The polyaffine problem is formulated as finding a feasible solution with a new constraint that preserves the affine trajectories of each local region. The natural solution uses a time-varying weight function, which is time and memory consuming in implementation. Our approach instead uses a composition of one or more diffeomorphisms of stationary velocities and is thus both efficient in computation and accurate in preserving the new constraints.

The key in constructing our weight function is to use the trajectory of each local region, instead of the region itself, to define the fusion weight for each local transform. In the case [9] when only the region is used in defining the weight, the affine velocity is only preserved at time $t = 0$. This may lead to inaccuracy in meeting the constraints. In contrast, our new approach preserves the affine velocity for any time t, and therefore preserves the affine transform.

Depending on the values of input affine transform and their regions, our approach requires one or more stationary velocity fields to preserve different affine velocities in each region over time. The number of stationary velocity fields are determined by the collision detection of region trajectories when collapsing them along the temporal direction. One real-life analogy to such a scheme is the traffic lights in an intersection. When two trajectories are overlapped at the intersection, a traffic light will indicate when the points along one direction should stop to allow the points along the other to pass the intersection, which solves the ambiguity in defining the velocity function at the intersection. Note that depending on the input affine transforms, there might be no collision and just one stationary velocity is enough.

References

1. Jenkinson, M., Smith, S.: A global optimisation method for robust affine registration of brain images. Medical Image Analysis 5(2), 143–156 (2001)
2. Thirion, J.P.: Image matching as a diffusion process: an analogy with maxwell's demons. Medical Image Analysis 2(3), 243–260 (1998)
3. Beg, M.F., Miller, M.I., Trouv, A., Younes, L.: Computing large deformation metric mappings via geodesic flows of diffeomorphisms. International Journal of Computer Vision 61(2), 139–157 (2005)
4. Rueckert, D., Sonoda, L., Hayes, C., Hill, D., Leach, M., Hawkes, D.: Nonrigid registration using free-form deformations: application to breast MR images. IEEE Transactions on Medical Imaging 18(8), 712–721 (1999)
5. Camion, V., Younes, L.: Geodesic interpolating splines. In: Figueiredo, M., Zerubia, J., Jain, A.K. (eds.) EMMCVPR 2001. LNCS, vol. 2134, pp. 513–527. Springer, Heidelberg (2001)
6. Ferrant, M., Warfield, S.K., Guttmann, C.R.G., Mulkern, R.V., Jolesz, F.A., Kikinis, R.: 3D image matching using a finite element based elastic deformation model. In: Taylor, C., Colchester, A. (eds.) MICCAI 1999. LNCS, vol. 1679, pp. 202–209. Springer, Heidelberg (1999)
7. Little, J., Hill, D., Hawkes, D.: Deformations incorporating rigid structures. Computer Vision and Image Understanding 66(2), 223–232 (1997)
8. Pitiot, A., Bardinet, E., Thompson, P.M., Malandain, G.: Piecewise affine registration of biological images for volume reconstruction. Medical Image Analysis 10(3), 465–483 (2006)
9. Arsigny, V., Commowick, O., Ayache, N., Pennec, X.: A fast and log-euclidean polyaffine framework for locally linear registration. Journal of Mathematical Imaging and Vision 33(2), 222–238 (2009)
10. Commowick, O., Arsigny, V., Isambert, A., Costa, J., Dhermain, F., Bidault, F., Bondiau, P.Y., Ayache, N., Malandain, G.: An efficient locally affine framework for the smooth registration of anatomical structures. Medical Image Analysis 12(4), 427–441 (2008)
11. Taquet, M., Macq, B., Warfield, S.K.: Spatially adaptive log-euclidean polyaffine registration based on sparse matches. In: Fichtinger, G., Martel, A., Peters, T. (eds.) MICCAI 2011, Part II. LNCS, vol. 6892, pp. 590–597. Springer, Heidelberg (2011)
12. Seiler, C., Pennec, X., Reyes, M.: Geometry-aware multiscale image registration via OBBTree-based polyaffine log-demons. In: Fichtinger, G., Martel, A., Peters, T. (eds.) MICCAI 2011, Part II. LNCS, vol. 6892, pp. 631–638. Springer, Heidelberg (2011)
13. Seiler, C., Pennec, X., Reyes, M.: Simultaneous multiscale polyaffine registration by incorporating deformation statistics. In: Ayache, N., Delingette, H., Golland, P., Mori, K. (eds.) MICCAI 2012, Part II. LNCS, vol. 7511, pp. 130–137. Springer, Heidelberg (2012)
14. Shepard, D.: A two-dimensional interpolation function for irregularly-spaced data. In: Proceedings of the 1968 23rd ACM National Conference, ACM 1968, pp. 517–524. ACM, New York (1968)