Eurographics Symposium on Geometry Processing (2007)
Alexander Belyaev, Michael Garland (Editors)

Dynamic Geometry Registration

Niloy J. Mitra* Simon Flöry* Maks Ovsjanikov‡ Natasha Gelfand§ Leonidas Guibas‡ Helmut Pottmann*

*Vienna University of Technology ‡Stanford University §Nokia Research Center

Abstract

We propose an algorithm that performs registration of large sets of unstructured point clouds of moving and deforming objects without computing correspondences. Given as input a set of frames with dense spatial and temporal sampling, such as the raw output of a fast scanner, our algorithm exploits the underlying temporal coherence in the data to directly compute the motion of the scanned object and bring all frames into a common coordinate system. In contrast with existing methods which usually perform pairwise alignments between consecutive frames, our algorithm computes a globally consistent motion spanning multiple frames. We add a time coordinate to all the input points based on the ordering of the respective frames and pose the problem of computing the motion of each frame as an estimation of certain kinematic properties of the resulting space-time surface. By performing this estimation for each frame as a whole we are able to compute rigid inter-frame motions, and by adapting our method to perform a local analysis of the space-time surface, we extend the basic algorithm to handle registration of deformable objects as well. We demonstrate the performance of our algorithm on a number of synthetic and scanned examples, each consisting of hundreds of scans.

Categories and Subject Descriptors (according to ACM CCS): I.3.5 [Computer Graphics]: Computational Geometry and Object Modeling.

1. Introduction

In recent years, significant advances have been made in developing high-speed shape acquisition devices. Using range scanning techniques such as structured light [RHHL02, KGG03, FB05] and spacetime stereo [DRR03, ZCS, WLG07], it is now possible to perform high-quality capture of detailed 3D geometry at close to video frame rates. As a result, it is now feasible to scan objects that are undergoing motion or deformation. The main limitation in scanning moving and deformable objects using slower methods, such as laser triangulation, has been the requirement that the object remains motionless during each scanning pass. This was not feasible, since each pass took a long time to cover the entire visible area. On the other hand, since each frame acquired by the fast scanning methods above covers a large visible area of the object at once, and the frames are closely spaced in time, it is now reasonable to assume that the scanned object remains stationary within each frame and the motion and deformation happen only between frames.

The challenge in registering data for high frame-rate scanners lies in processing the large amount of noisy geometric information produced by these devices. The ultimate goal of a scanning system is to produce a complete 3D model of the scanned object. However, the output produced by the ranging devices is usually an unstructured cloud of points that only partially covers the surface of the object. In each frame, the points are given in the reference system of the scanner, they are not registered in object space, and correspondences between points in different frames are rarely available.

While there has been a large amount of work on alignment of point clouds (see Section 2), the research until now has mostly focused on processing relatively small numbers of scans that are produced by the slower scanning systems. As a result, the traditional method of building models from range data is to align all pairs of overlapping scans independently, without taking into account how other scans in the systems are registered [LPC*00, BR02]. Since alignment errors tend to accumulate, before pairwise transformations can be chained together to integrate all scans into a common coordinate system, a relaxation is usually performed to spread out the accumulated error among all scans in the system [BSGL96, Pul99]. This standard approach to building 3D models works well when the number of input scans is relatively small and there is relatively little coherence between successive inter-frame motions — which is the case when each scan takes a long time, and the object or the scanner is manually re-positioned between each successive scan as to maximize the newly visible area.

However, when scanning objects at high frame rates, inter-frame motions are generally small, while the number of frames is now very large (often on the order of many hundreds or even thousands). Processing all pairs of overlapping frames, filtering away bad alignments, and performing global relaxation becomes too expensive. At the same time, if the object is undergoing some motion while it is being scanned (e.g. rotating so that all sides eventually become visible), the motion is likely to remain coherent over some number of frames.

In this paper, we propose an algorithm for registration of point clouds of moving and deforming objects. Instead of performing independent pairwise alignment between consecutive frames, as has been done in previous methods, we exploit the underlying temporal coherence in the data to integrate together data from several frames at once and directly compute the object motion from the raw scanner output *in one shot*. Simply put, we add a time coordinate to all input points based on their respective frames, and pose the problem of computing the motion of each scan as an estimation of certain surface properties of the resulting kinematic space-time surface. The main advantage of our algorithm over prior methods is that we do not have to explicitly compute corresponding points between successive frames; instead, we compute the alignment by exploiting the underlying smoothness of the space-time surface which results from the property that the inter-frame motions remain coherent.

The main contributions of this work are as follows:

- We describe an algorithm for computing all inter-frame motions of a set of scans using fundamental kinematic properties of the space-time surface formed by the input points. The method does not require establishing correspondence across different scans.
- We analyze the performance of our algorithm with respect to increasing amounts of noise and inter-frame displacement. We also compare its performance with that of ICP [BM92, CM92], which is the standard method for pairwise registration. We show that since the local neighborhoods on the space-time surface include data from multiple frames at once, our method, under sufficient time sampling, is not as susceptible to the problem of accumulating errors as the pairwise alignment methods.
- To handle sparse sampled data, we propose two refinements to our basic algorithm to improve the motion estimates, at the cost of minor increase in time complexity.

2. Related Work

Range image registration is an integral part of most shape acquisition pipelines and a large number of algorithms has been developed to address this challenging problem. In general registration algorithms are usually classified into pairwise and multi-view registration. Pairwise registration finds aligning transformations between overlapping pairs of scans, and these transformations can then be chained together to compute the final object-space pose for each scan. Since only pairs of scans are considered independently, local errors tend to accumulate, causing poor alignments especially between the scans with many intermediate transformations. Thus, multi-view registration is used to spread the accumulated error among all scans in the system.

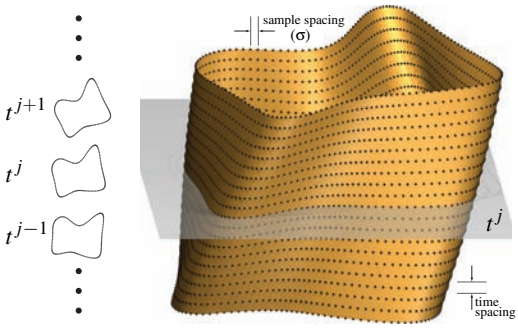

Figure 1: *Space-time Surface.* Kinematic space-time surface traced by a rigidly moving 2D curve. For scans of a moving 3D object the corresponding surface resides in 4D.

Pairwise registration algorithms for range data are usually based on finding sets of corresponding points between the two input scans, which are then used to compute the aligning transformation. Direct methods find correspondences by comparing local shape descriptors computed on the inputs [HH03, GMGP05, LG05]. Iterative methods, such as the classic Iterated Closest Point (ICP) algorithm [BM92, CM92] and its variants [RL01] are often used when the relative transformations between the two scans are not very large. These methods alternate the computation of correspondences with refining the aligning transformation, and almost always require multiple iterations before the final alignment is found [MGPG04].

As mentioned, the pairwise transformations are refined by multi-view registration algorithms that try to aggregate data from several scans and distribute the accumulated error. Multi-view methods are usually iterative in nature, since they perform an optimization over all transformations computed by the pairwise method [Pul99, SLW02, KLMV05].

The methods described above deal with rigid registration. When the scanned object undergoes deformation between the successive scans, the registration problem becomes much more difficult due to the larger number of parameters that need to be recovered. Pairwise registration methods based on extending ICP to include deformations [HTB03], thin-plate splines [BR07], and optimization over correspondence space [ASK*05] have been proposed. These methods are correspondence-based and are generally aimed at aligning a small number of deformable objects or range images.

When the scanning speed approaches video frame rates, building a 3D model can be thought of not as a problem in pairwise alignment or registration, but as object tracking.

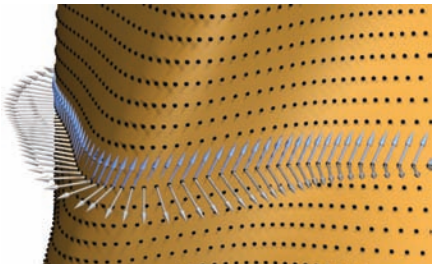

Figure 2: *Space-time Velocity Vectors.* The instantaneous velocity vectors (in blue) are tangential to the kinematic space-time surface generated by rigidly transforming shapes. The space-time normals are shown in gray.

The input comes as a dense stream of *frames*, and the relative motions between successive pairs of frames are likely to be similar, if the scanning speed is fast relative to the object motion. The idea of exploiting inter-frame coherence is extensively used for object tracking in computer vision [BB95] with techniques like optical flow, but is only recently started to be used in 3D registration, e.g. in [RHHL02].

The main distinction between our algorithm and prior work on rigid and non-rigid registration is that instead of establishing correspondence between points from different scans, we use kinematic properties of space-time surfaces to solve for aligning motion. In addition, instead of processing scans in pairs, we aggregate together information from multiple nearby frames when computing aligning transformations. As a result, when time sampling is sufficient, our algorithm does not require the multi-view error distribution step. Our approach is particularly well suited for applications of high frame rate range scanning which produce a large number of frames (on the order of thousands in some examples) with temporally coherent inter-frame motions.

3. Space-time Registration

3.1. The Space-time Surface

We describe our space-time registration framework by considering the following basic setup. An object is undergoing some smooth motion $\alpha(t)$. We are given a set of frames (e.g. range images) $\{P^0, P^1, \ldots, P^n\}$ each consisting of $n_j = |P^j|$ points $\mathbf{p}_1^j, \mathbf{p}_2^j, \ldots, \mathbf{p}_{n_j}^j$. Each frame P^j corresponds to a potentially different view of the moving object, acquired at time t^j. The coordinates of the points in each frame are given in the local coordinate system of the scanner, and the correspondences between points in different frames are unknown. We assume that the acquisition process is fast enough that all motion happens between successive frames, and for now we assume that the object is moving rigidly. The goal of our multi-frame alignment algorithm is to register all frames into a common coordinate system (usually that of P^0). We accomplish this by first computing a set of n inter-frame transformations $\alpha_j := (\mathbf{R}_j, \mathbf{t}_j)$, such that α_j aligns P^j to P^{j+1}, and then suitably applying the computed transforms to obtain the final pose of each frame.

We change to a space-time model as follows: Say frame P^j is acquired at a time t^j. We treat this scanning time as an additional coordinate for all points in P^j. That is a space-time frame, or *time slice*, and is defined as $\widetilde{P}^j \equiv \{\widetilde{\mathbf{p}}_i^j\} := \{(\mathbf{p}_i^j, t^j), \mathbf{p}_i^j \in \mathbb{R}^d, t^j \in \mathbb{R}\}$, where $d = 2$ for curves and $d = 3$ for surface alignment. We will use the $\widetilde{\ }$ notation to differentiate points, normals, and frames in the space-time domain from their regular (spatial) counterparts. The points in $\widetilde{P}^0, \ldots, \widetilde{P}^n$ now lie on (but, due to acquisition errors, usually just close to) a d-dimensional kinematic space-time surface $S \subset \mathbb{R}^{d+1}$ [PW01] (see Figure 1 for a curve example, $d = 2$). It turns out that we can extract the parameters of inter-frame motions α_j using the fundamental kinematic properties of S.

By construction, for each time slice $\widetilde{P^j}$ there exists a $(d+1)$-dimensional motion $\widetilde{\alpha_j}$ that transforms each point $\widetilde{\mathbf{p}}_i^j$ so that $\widetilde{\alpha_j}(\widetilde{\mathbf{p}}_i^j)$ still lies on the space-time surface. This motion consists of the d-dimensional unknown rigid motion $(\mathbf{R}_j, \mathbf{t}_j)$ that aligns the consecutive frames in the space domain, and a translation by $\Delta t^j := t^{j+1} - t^j$ along the time axis,

$$\widetilde{\alpha_j}(\widetilde{\mathbf{p}}_i^j) = \left(\mathbf{R}_j \mathbf{p}_i^j + \mathbf{t}_j, t^j + \Delta t^j\right). \tag{1}$$

If we knew the correspondences between the points $\mathbf{p}_i^{j+1}$ and $\mathbf{p}_i^j$, we could compute the motion parameters directly (and indeed would not even need the space-time formulation) by minimizing the distances between corresponding points [ELF97]. In the absence of correspondences, all we know is that each transformed point $\widetilde{\alpha_j}(\widetilde{\mathbf{p}}_i^j)$ should lie somewhere on the space-time surface S. Therefore, if we assume that all points in any time slice undergo the same motion (rigid motion assumption), then the optimal $\widetilde{\alpha_j}$ should satisfy:

$$\widetilde{\alpha_j} = \operatorname{argmin} \sum_{i=1}^{|P^j|} d^2(\widetilde{\alpha_j}(\widetilde{\mathbf{p}}_i^j), S). \tag{2}$$

To compute the distance between the transformed points and the space-time surface, we approximate S by its space-time tangent plane at $\widetilde{\mathbf{p}}_i^j$. Then, the distance between the transformed points and the space-time surface in Equation 2 is

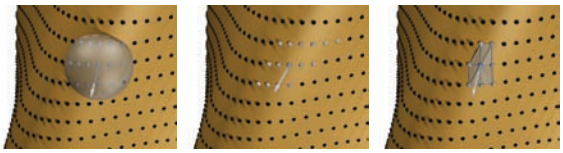

Figure 3: *Space-time Normal Estimation.* (Left) Weighted PCA based normal estimation with uniform neighborhood, (center) updated neighborhood after one step of iterative normal refinement using initial instantaneous velocity estimates, and (right) normal estimation using local triangulation (local tetrahedralization for 3D scans).

Figure 4: *Reconstruction Quality.* Reconstruction from 300 *simulated* noisy scans (point clouds) of the Stanford bunny. In the center, we overlap the reconstructed model with the original model to show error due to dynamic registration. No global error distribution, or data smoothing was performed.

minimized when each point $\widetilde{\mathbf{p}}_i^j$ is moved tangentially to S. That is, the instantaneous velocity vector field of the rigid motion $\widetilde{\alpha_j}$ that aligns consecutive frames should be perpendicular to the normal field of S (see Figure 2). This kinematic constraint [PW01] is a fundamental property of any space-time surface generated by a rigid motion. This property allows us to solve the registration problem as a kinematic minimization problem on the space-time surface. We will first recover the $(d+1)$-dimensional velocity vectors that minimize the tangential displacement at each point of S, and then use these velocity vectors to compute the d-dimensional transformations that register together the input frames.

The velocity vector field of the $(d+1)$-dimensional rigid motion $\widetilde{\alpha_j}$ can be decomposed into velocity in space and in time. The velocity in time is constant by construction, since the frames are translated along the time axis to form the space-time surface (assuming that the scanner operates at a constant frame rate). The velocity in space is the velocity vector field of the underlying aligning rigid motion α_j.

It is well known [BR90] that the instantaneous velocity vector field of any 3-dimensional rigid motion is linear, and can be expressed as:

$$\mathbf{v}(\mathbf{p}_i^j) = \mathbf{c}_j \times \mathbf{p}_i^j + \bar{\mathbf{c}}_j. \tag{3}$$

Therefore, the velocity vector field in space-time is given as $\widetilde{\mathbf{v}}(\widetilde{\mathbf{p}}_i^j) = (\mathbf{c}_j \times \mathbf{p}_i^j + \bar{\mathbf{c}}_j, 1)$. Later in Section 4 we justify the choice of the unit time scale. The unknown parameters $(\mathbf{c}_j, \bar{\mathbf{c}}_j)$ are such that the velocity vectors lie in the tangent space of S at each $\widetilde{\mathbf{p}}_i^j$, and can be found by minimizing:

$$\min_{\mathbf{c}_j, \bar{\mathbf{c}}_j} \sum_{i=1}^{|P^j|} w_i^j \left[(\mathbf{c}_j \times \mathbf{p}_i^j + \bar{\mathbf{c}}_j, 1) \cdot \widetilde{\mathbf{n}}_i^j \right]^2, \tag{4}$$

where $\widetilde{\mathbf{n}}_i^j$ is the normal to the space-time surface S at the point $\widetilde{\mathbf{p}}_i^j$ and w_i^j is the weight of the contribution of the given point to the minimization (to be defined in Section 3.2).

Equation 4 gives the fundamental relationship between the space-time velocity vectors of the aligning inter-frame motion and the normals to the space-time surface S. To solve for the parameters of the velocity field at each time slice, we differentiate with respect to $(\mathbf{c}_j, \bar{\mathbf{c}}_j)$, which yields a system of linear equations of the form,

$$A\mathbf{x} + \mathbf{b} = 0. \tag{5}$$

Writing the space-time normal as $\widetilde{\mathbf{n}}_i^j = (\bar{\mathbf{n}}_i^j, n_i^j)$, $\bar{\mathbf{n}}_i^j \in \mathbb{R}^d$ and $n_i^j \in \mathbb{R}$, these components are given by:

$$A = \sum_{i=1}^{|P^j|} w_i^j \begin{bmatrix} \bar{\mathbf{n}}_i^j \\ \mathbf{p}_i^j \times \bar{\mathbf{n}}_i^j \end{bmatrix} \begin{bmatrix} \bar{\mathbf{n}}_i^{j\,T} & (\mathbf{p}_i^j \times \bar{\mathbf{n}}_i^j)^T \end{bmatrix},$$

$$\mathbf{b} = \sum_{i=1}^{|P^j|} w_i^j n_i^j \begin{bmatrix} \bar{\mathbf{n}}_i^j \\ \mathbf{p}_i^j \times \bar{\mathbf{n}}_i^j \end{bmatrix}, \text{ and } \mathbf{x} = \begin{bmatrix} \bar{\mathbf{c}}_j \\ \mathbf{c}_j \end{bmatrix}.$$

Remark 1. Notice that the space-time surface is generated by a continuous $(d+1)$-dimensional motion of a profile curve ($d=2$) or surface ($d=3$). At points in a time slice, the velocity vectors are tangential to this surface, and thus are the same as those of a kinematic surface [PW01]. Kinematic surfaces are shapes for which there exists a continuous rigid motion such that the transformed surface is everywhere in contact with the original surface. Equation 4 is similar to an equation used to determine if a set of points was sampled from some kinematic surface [PW01].

Remark 2. Equation 4 is also similar to the motion constraint equation used in computer vision for computation of optical flow in images [Fau90, BB95]. The image gradients that are used in optical flow are replaced in our framework by the normal field of the space-time surface, while the image flow corresponds to the velocity vector field of the aligning rigid motion in our space-time registration framework. When the input frames themselves come from a kinematic surface (for example, when scanning a ball or a plane), A is not full rank, and Equation 4 has multiple solutions. This implies there are multiple valid transformations that can register the frames equally well, analogous to the aperture problem in optical flow.

3.2. Normal Estimation

From the above derivation we see that the crucial step in computing the velocity vectors is the estimation of normals for points on the space-time surface. We can compute the normal field by locally fitting a hyper-plane in $\mathbb{R}^{d+1}$ to the space-time neighborhood around each point $\widetilde{\mathbf{p}}_i^j$ using the standard PCA technique [MNG04]. In addition, we can use the shape of the covariance ellipsoid as our confidence in the quality of the plane fit: With $\lambda_1 \leq \cdots \leq \lambda_{d+1}$ as eigenvalues of the covariance matrix, we assign the per-point weights in Equation 4 as $w_i^j := \exp(-\lambda_1 / \sum_{k=1}^{d+1} \lambda_k)$ as proposed by Pauly et al. [PGK02].

To compute the plane fit, we first need to form a space-time neighborhood $N_{r_n}(\widetilde{\mathbf{p}}_i^j) := \{\widetilde{\mathbf{q}} : \|\widetilde{\mathbf{p}}_i^j - \widetilde{\mathbf{q}}\| \leq r_n\}$ for each point $\widetilde{\mathbf{p}}_i^j$ in the current time slice. There are two issues that need to be addressed when computing this neighborhood:

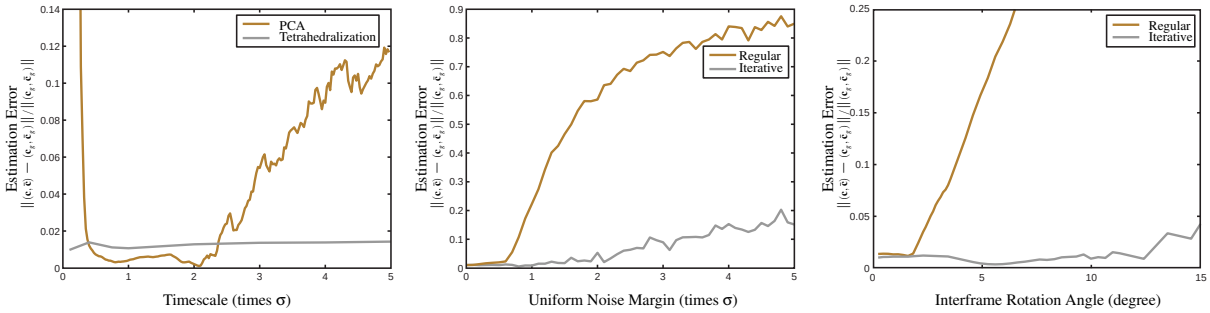

Figure 5: *Effects of Time Scaling, Uniform Noise, and Interframe Motion.* (Left) Under low noise, local tetrahedralization reliably approximates local space-time surface even with increasing time scaling. In contrast, PCA is reliable only when spacing and temporal sampling are comparable. (Center) Under increasing noise margin, iterative normal refinement performs better than normal PCA at the cost of increased computation. (Right) Similar behavior is observed for increasing interframe motion. All plots were generated using 300 simulated scans of the Stanford bunny. Estimation error was measured with respect to ground truth motion parameters $(\mathbf{c}_g, \bar{\mathbf{c}}_g)$.

the relative scaling between the space and time coordinates of the points, and the magnitude of the radius r_n.

To ensure that the neighborhood $N_{r_n}(\widetilde{\mathbf{p}})$ introduces little bias between the points from the current time slice and points from the neighboring time slices, we set the inter-frame spacing along the time axis to be equal to the average sample spacing (σ) of the input frames. This sample spacing can be determined from the properties of the scanning device, or estimated directly from the data [MNG04]. Although the spatial density can vary over a large number of time slices, locally in time they can be assumed constant, and stacked as mentioned above.

Given comparable sampling in both space and time, we now need to ensure that the size of the neighborhood is large enough so that the plane fit is not impacted by noise, while at the same time is not too large that curvature of the surface affects the plane fit. The quality of the plane fit can be estimated using the eigenvalues given by the PCA. Therefore, we gradually increase the radius of the neighborhood r_n until the eigenvalues stabilize, as proposed in [PGK02, MNG04]. For our experiments, maximum allowed r_n was 5σ. We used approximate nearest neighbor structure ANN [MA97] to perform neighborhood search in 4D – more than 80% of the total computation time was used for neighborhood query processing.

Missing data due to self-occlusions in the input scans create holes in our space-time surface. Using fixed radius neighborhoods usually results in lower quality space-time normals being computed near these boundary points. In our case, since the velocity vectors are computed for entire frames, points from areas with good normals dominate and our algorithm still computes good velocity estimates for each frame in the presence of such incomplete data. Although we did not find it necessary, it is also possible to perform boundary detection on the space-time surface using one of the standard methods [RL01, PMG*05].

3.3. Registration Algorithm

The discussion above now gives us all the components for a multi-frame registration algorithm. Given as input frames $P^0, P^1, \ldots, P^n$ the algorithm produces the parameters $(\mathbf{c}_j, \bar{\mathbf{c}}_j)$ that give the velocity vectors between consecutive frames.

To produce the final registered model, we need to convert the instantaneous velocities of each point into inter-frame rotation and translation. From spatial kinematics [PW01] we know that this motion is a rotation about an axis A_j by an angle ρ_j and a translation parallel to A_j by a distance of $h_j = p_j\rho_j$. The values ρ_j, p_j and the Plücker coordinates of the axis A_j (direction vector $\mathbf{g}_j$ and momentum vector $\bar{\mathbf{g}}_j = \mathbf{y}_j \times \mathbf{g}_j$, where $\mathbf{y}_j$ is any point in A_j) are given by:

$$\mathbf{g}_j = \frac{\mathbf{c}_j}{\|\mathbf{c}_j\|}, \; \bar{\mathbf{g}}_j = \frac{\bar{\mathbf{c}}_j - p_j\mathbf{c}_j}{\|\mathbf{c}_j\|}, \; p_j = \frac{\mathbf{c}_j \cdot \bar{\mathbf{c}}_j}{\mathbf{c}_j^2}, \; \rho_j = \|\mathbf{c}_j\|. \quad (6)$$

To summarize, the algorithm proceeds as follows:

1. Form the space-time points by adding a time coordinate $t^j = \sigma \cdot j$ to the scan points in each frame P^j, where σ is the average sampling density of the input frames. Since most scanners operate at a constant frame rate, this uniform scaling of the time coordinate is justified. This scaling is done to make sure that neighborhood $N_{r_n}(\widetilde{\mathbf{p}})$ used in normal estimation has similar density of points in space and in time. In Section 4, we show that the motion estimates are invariant to local changes of this time scale.
2. Form the space-time neighborhood $N_{r_n}(\widetilde{\mathbf{p}})$ around each data point $\widetilde{\mathbf{p}}$, and estimate the normals $\widetilde{\mathbf{n}}_i^j$ using PCA.
3. For each time slice P^j, form the minimization in Equation 4 and solve for parameters $(\mathbf{c}_j, \bar{\mathbf{c}}_j)$.
4. Compute the transformation parameters for time slice t^j from the velocity field with help of Equation 6. Using quarternions, the computed parameters are converted to the transform $(\mathbf{R}_j, \mathbf{t}_j)$ and suitably applied to produce the registered model.

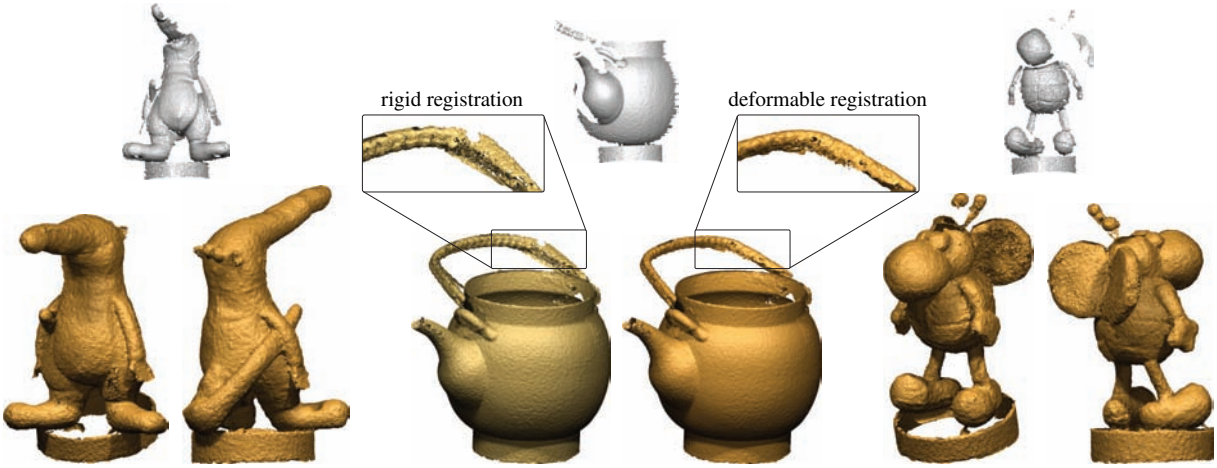

Figure 6: *Scan Assembly.* Reconstruction results on scans of coati, teapot, and bee models scanned using a 17 Hz range scanner. Each model was reconstructed from 2,200 scans registered using regular scheme on adjacent frames. No global error distribution, or noise smoothing was applied. The handle of the teapot moved during scanning, resulting in poor registration (left teapot). The right teapot shows improved alignment when our deformable registration framework was used.

Results. Figure 4 shows the result of our algorithm on synthetic scans of the bunny model. We generated 300 range images by simulating a scanner as the model was rotated by 1.5 degrees between successive frames. As our simulated scanner, we used the z-buffer to generate simulated range images, to which uniformly distributed zero-mean noise was added. We then computed transformations α_j between adjacent frames, and using them registered all the frames back to the first scan. We evaluated the reconstruction quality by comparing with the original model.

We tested our algorithm on data acquired using a stereo and active illumination based 3D scanner developed by Weise et al. [WLG07]. The scanning rate is 17 Hz. For all our examples, no data smoothing or global error distribution was performed — we worked directly on the raw scanned point clouds. Figure 6 shows our results for coati, teapot, and bee model. For each example, 2,200 scans were registered together by combining motion between adjacent frames in time. The handle of the teapot moved during the scanning process, resulting in a distorted handle in the rigidly registered model. Later in Section 5, we address this issue.

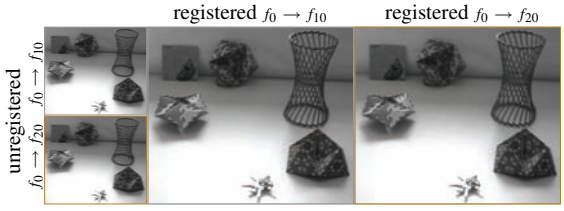

Figure 7: *Image Registration.* Application of dynamic registration for video stabilization. The sequence was acquired using a point and shoot digital camera that was rotated and moved around during acquisition. Using gray scale values as depth, we solve for the 3D camera motion rather than a rigid transform in the image plane.

We applied our method to another type of input — a video sequence acquired using a point and shoot digital camera. Using the grayscale values, each frame was converted to a 3D point cloud (height field in this case), and our method applied (see [BR07] for a similar application). Observe that we solve for the 3D camera motion rather than a rigid transform in the image plane. Figure 7 shows our results, which are similar to using optical flow for this problem [BB95].

4. Analysis of the Basic Algorithm

Effect of Time Scaling. Now we examine how the different parts of the registration algorithm are affected by the time scaling. Let the time axis scale by some λ, thus we can scale $t^j \mapsto \lambda t^j$. Then, the velocity vector field becomes $\tilde{\mathbf{v}}((\mathbf{p}_i, \lambda t^j)) = (\bar{\mathbf{c}} + \mathbf{c} \times \mathbf{p}_i, \lambda)$ and the space-time normal $\tilde{\mathbf{n}}_i = (\bar{\mathbf{n}}_i, n_i^d)$ turns out to be $(\lambda \bar{\mathbf{n}}_i, n_i^d)/C_i$, where the normalizing constant $C_i = (\lambda^2 ||\bar{\mathbf{n}}_i||^2 + (n_i^d)^2)^{1/2}$. Inserting these scaled expressions into Equation 4 results in w_i^j getting scaled by λ^2/C_i^2. However, observe that for small motions, points do not undergo large transformations in space, but necessarily undergo a λ translation in time. This means that the 4D normal at any point is dominated by the first three components, and hence $|n_i^d| \approx \varepsilon$ and $\|\bar{\mathbf{n}}_i\| \approx 1 - \varepsilon$, for small ε. Furthermore, because the normal field is smooth, the constants $\lambda^2/{C_i}^2$ are approximately equal. It follows that the method for computing the motion parameters $(\bar{\mathbf{c}}_j, \mathbf{c}_j)$ is insensitive to local changes in λ. However, this is only true if the normal estimation can capture $(\bar{\mathbf{n}}_i, n_i^d) \mapsto (\lambda \bar{\mathbf{n}}_i, n_i^d)/C_i$ due to scaling along the time axis.

The regular PCA method has this desired property pro-

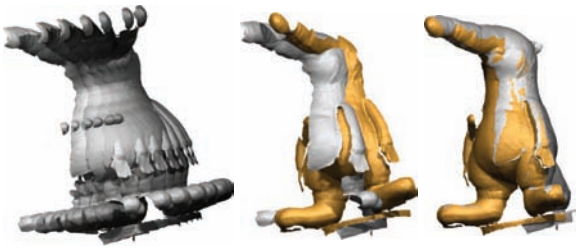

Figure 8: *Accumulation Error.* (Left) 300 scans of the coati model were registered (adjacent) pairwise, and the computed transforms were used to align the last frame to the first frame. Dynamic registration (right) gives good alignment, while ICP algorithm results in accumulation error (center). In presence of sufficient time sampling, our method does not need global error distribution.

vided that the set of neighbors chosen for the plane fit is same. Due to scaling of the fourth coordinate of every point, the fourth coordinate of the eigenvector of the covariance matrix scales inversely, which implies that PCA will correctly capture the effect of time scaling on normals – as long as the set of neighbors is same. Such local time invariance is observed in Figure 5 (left) for time scales between 0.5 and 2, using PCA based normal estimates. Outside this range, the neighborhood set starts changing too drastically, and hence the normal scaling assumption no longer holds. Local tetrahedralization (Section 5), under similar conditions, exhibits a wider stable range of time-scale invariance (Figure 5).

Sensitivity to Noise. In order to examine stability issues of dynamic geometry registration using the space-time model, we consider the effect of noise in the input data on the final motion estimates $\mathbf{x} := (\bar{\mathbf{c}}_j, \mathbf{c}_j)$. Let the noise in points be bounded by $\|\varepsilon_0\|$, and that in normals by $\|\varepsilon_1\|$. Please note that for PCA, the relation between $\|\varepsilon_0\|$ and $\|\varepsilon_1\|$ has been studied [MNG04]. We obtain $\|A'\|_F = \|A\|_F + o(\|\varepsilon_0\|) + o(\|\varepsilon_1\|)$ and $\|\mathbf{b}'\| = \|\mathbf{b}\| + o(\|\varepsilon_0\|) + o(\|\varepsilon_1\|)$ for A and $\mathbf{b}$ in Equation 5, where $\|.\|_F$ denotes the Frobenius norm. From the well known stability results [IK66] for the solution of such a linear system,

$$\frac{\|\mathbf{x}-\mathbf{x}'\|}{\|\mathbf{x}\|} \leq \frac{k(A)}{1-k(A)\frac{\|A'-A\|_F}{\|A\|_F}} \left(\frac{\|A'-A\|_F}{\|A\|_F} + \frac{\|\mathbf{b}'-\mathbf{b}\|}{\|\mathbf{b}\|} \right),$$

it follows that the relative error in $\mathbf{x}$ is linear in $\|\varepsilon_0\|$ and $\|\varepsilon_1\|$ as well. The constant may be large for bad condition number $k(A)$ (e.g. when we solve for nearly slippable motions).

Relation to ICP. An alternative for computing inter-frame transformations is to register pairs of consecutive scans using a correspondence search based method, such as ICP [BM92, CM92]. The main advantage of our method over pairwise registration is that information from multiple frames is incorporated into the motion computation. This generates a more globally consistent velocity field. Pairwise methods usually have to perform a smoothing or global relaxation [Pul99] to spread the accumulated error over multiple frames. This is not necessary in our method.

Figure 8 shows the results of applying pairwise ICP registration to a set of 300 scans and chaining pairwise transformations to align the first and last scans. Notice that there is a sizeable alignment error. When our method is applied to the same input, the resulting alignment is correct. Compared to our single pass algorithm, ICP required an average of 6 iterations to converge for each pairwise scan alignment.

We can also compare the objective function in Equation 4 computed by our algorithm to the objective function of ICP. Consider Equation 2 which gives us the estimate of the optimal motion. As mentioned, we approximate the space-time surface by its tangent plane at point $\widetilde{\mathbf{p}}_i^j$. Another alternative would be to approximate the space-time surface by its tangent plane at the footpoint $\widetilde{\mathbf{f}}$, the point in the $(j+1)$st frame whose spatial coordinates are closest to those of $\widetilde{\mathbf{p}}_i^j$. Then, by definition the transformation $\widetilde{\alpha_j}$ uniformly translates points in time by 1, so the time coordinates of $\widetilde{\mathbf{f}}$ and $\widetilde{\alpha_j}(\widetilde{\mathbf{p}}_i^j)$ are the same. Thus, the distance between the tangent plane at $\widetilde{\mathbf{f}}$ and the transformed point $\widetilde{\alpha_j}(\widetilde{\mathbf{p}}_i^j)$ gives:

$$\begin{aligned} d(S, \widetilde{\alpha_j}(\widetilde{\mathbf{p}}_i^j)) &\approx [\widetilde{\mathbf{f}} - \widetilde{\alpha_j}(\widetilde{\mathbf{p}}_i^j)] \cdot \widetilde{\mathbf{n}}_{\mathbf{f}} \\ &= [\mathbf{f} - \alpha_j(\mathbf{p}_i^j)] \cdot \bar{\mathbf{n}}_{\mathbf{f}} \end{aligned}$$

where, $\bar{\mathbf{n}}_{\mathbf{f}}$ represents the spatial coordinates of the space-time normal $\widetilde{\mathbf{n}}_{\mathbf{f}}$ at the footpoint. The normal to the curve or surface (time slice) at $\mathbf{f}$ is in the direction of $\bar{\mathbf{n}}_{\mathbf{f}}$. Therefore, if the distance to the space-time surface is approximated in this manner, the objective function of Equation 2 is equal to the objective function of a weighted point-to-plane ICP method [CM92]. However, in the case of uniform rigid motion, the approximation introduced in our method is more accurate because instead of considering point-plane pairs which only gives a second order approximation to the distance field, it permits us to rephrase the optimization from the point of view of instantaneous motion, which is *locally exact*. Thus, the only error in our algorithm comes from incorrect normal estimation, or the uniform rigid motion assumption not being fulfilled.

5. Modification of the Basic Algorithm

In this section, we describe extensions of the basic space-time registration method to deal with larger inter-frame motions, articulated motion, and general deformations.

Iterative Normal Refinement. In Section 4 we evaluate our algorithm in the presence of sufficient time sampling. However, the performance suffers in the absence of enough time slices. This is to be expected, since we have no knowledge of correspondences between points across different frames, which would allow us to deal with larger motions.

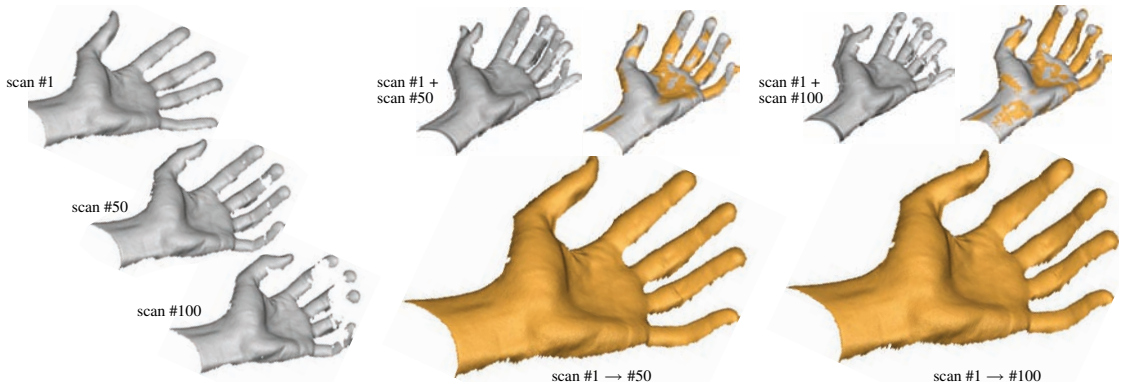

Figure 9: *Deformable Body Registration.* Deformable dynamic registration of 100 scans of a deforming hand. The first frame is deformed using the computed motion. In presence of reasonable data, we get good tracking. However, the method fails gracefully (right) when major parts of the data are missing (Data around the fingers are consistently missing in slices $50-100$).

We can mitigate this problem at the cost of increased computation time by using the initial velocity estimates to improve neighborhood selection for the normal estimation. The iterative normal refinement procedure works as follows:

1. Compute initial velocity estimate for each frame as described in Section 3.3.
2. The current motion estimates are converted to displacements using Plücker coordinates, and used to reselect the set of neighbors. Intuitively this deforms the initial spherical neighborhood according to the current motion estimate to better approximate the local surface (Figure 3).
3. Re-compute the normals using the new neighborhood and, re-estimate the motion parameters.
4. Iterate steps 2 and 3 until the motion estimates stabilize.

Although the iterative nature of this method makes it similar to ICP, there are two important differences. First, the method works without ever computing point correspondences. The information about the aligning motions is derived from the point neighborhoods in the space-time surface. Second, the input points are never moved, we are not combining results of several iterations to get the final position, as is done in ICP. Instead, in each step, using results from previous iteration, we improve estimates of normal and velocity vectors in the current step.

Figure 5 compares the performance of regular and iterative normal refinement schemes on simulated scans of the Stanford bunny with increasing noise data, and increasing inter-frame rotation, respectively. Relative motion estimation error compared to the known ground truth is plotted. For high interframe motion, as high as 15 degrees, the iterative scheme (5-6 steps) still results in low error, while the regular scheme performs poorly. The iterative scheme significantly relaxes the time sampling requirement, but at the cost of increased computation time. However, in presence of enough time slices, as in our examples when using the structured light scanner of [WLG07], the regular scheme is the method of choice due to its speed with negligible loss in accuracy.

Normals from Local Surface Approximation. In Section 3.2, to correctly estimate space-time normals using local PCA, we require sufficient scanning frequency such that space and time spacings are comparable. However, in low noise condition, we can reliably estimate normals using a local surface meshing approach as follows: For space-time surfaces traced by curves, around each vertex $\tilde{\mathbf{p}}_i^j$ we perform a local surface triangulation (in 3D). Then to estimate a normal at the vertex, we average its one-ring face normals. Observe that the one ring neighbor of a vertex contains points from the adjacent time slices. Hence to construct a local triangulation we only consider neighbors of a vertex in current time slice, and also neighbors from adjacent time slices, t^{j-1} and t^{j+1}. Since our local surface is only for normal estimation, we neglect global issues like intersecting triangles. Further our optimization being oblivious to orientation of normals, we do not require a consistent normal orientation.

For surfaces, we generalize this approach to generate a local tetrahedral mesh from 4D space-time data. Again for a vertex $\tilde{\mathbf{p}}_i^j$, we take its neighbors from time t^{j-1}, t^j, and t^{j+1}. We now want to construct local tetrahedra with $\tilde{\mathbf{p}}_i^j$ as a vertex. We consider three types of tetrahedra: Tetrahedra formed using a triangular face in slice t^j and the closest neighbor from adjacent time slice; ones formed by a triangular face in an adjacent slice and $\tilde{\mathbf{p}}_i^j$, and those spanned by an edge in t^j and an edge in t^{j+1} (or t^{j-1}). Again we do not strive to compute a globally consistent tetrahedral mesh. To compute a vector orthogonal to three linearly independent edges of a tetrahedron, we use the four dimensional variant of the cross product. Combining such normals from all the constructed tetrahedrons pivoted at $\tilde{\mathbf{p}}_i^j$, we estimate its normal. Such normals better capture the surface behavior even with large time scaling in presence of low noise (see Figure 5). Such local tetrahedralization can theoretically lead to arbitrarily bad estimates at isolated points, but since our final optimization is over a large set of points, our estimation is ro-

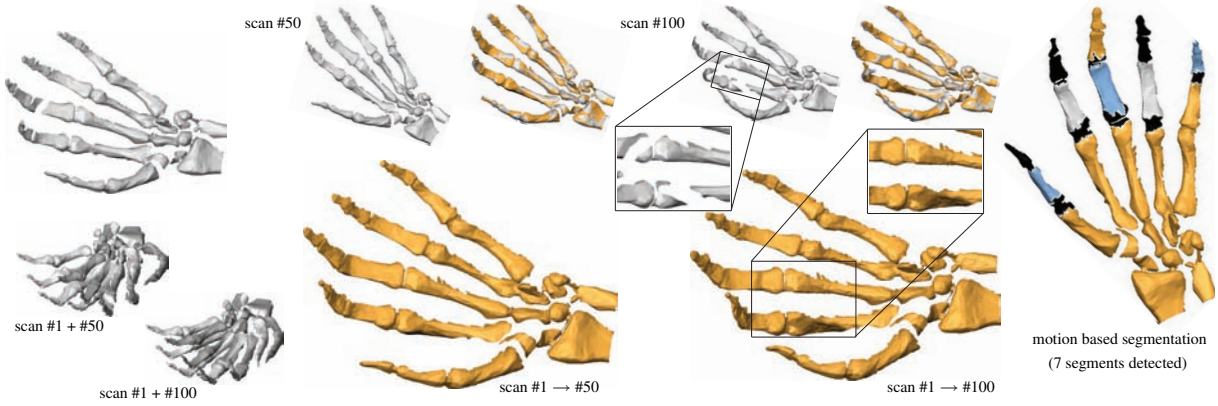

Figure 10: *Articulated Motion.* Non-rigid registration between adjacent pairs of 100 simulated scans from a closing skeleton hand with simultaneous 'scanner motion'. Using the computed motion, we deform the first scan – we show the results at 50-th and 100-th frames. Scan completion is achieved by transferring data from first frame onto others. (Right) The object can be segmented using a clustering in 6D space of $(\mathbf{c}, \bar{\mathbf{c}})$. We detected 7 dominant segments. The remaining object is in black.

bust to a few such outliers. To increase robustness (e.g. near silhouettes edges or when motion smoothing is required), instead of just adjacent time slices, more neighboring ones can be considered at the cost of increased complexity.

Deformable Bodies. Deformable bodies even when scanned with sufficient time density, do not satisfy our requirement that the adjacent frames are related by rigid transforms. The resulting error quickly accumulates over a sequence of frames. However, it is possible to apply our algorithm to this scenario with slight modifications. Observe that most deformations have low degrees of freedom and, hence locally (in space) the deformations are highly correlated [ACP02] and can be considered rigid. Using this intuition, we perform a kinematic minimization as in Equation 4, but instead of globally over an entire time slice, locally in overlapping regions within a time slice. In each time slice, our algorithm proceeds as follows:

1. Generate a set of roughly uniformly spaced sampled centers from the current time slice data (3D point cloud) using a method similar to [Tur92]. For all our results, the sample-center spacing was chosen to be 10σ.
2. For each such sample-center, perform kinematic minimization as in Equation 2 considering only points from the current time slice that lie within a local neighborhood of roughly sample-center spacing (10σ).
3. For each sample-center, compute the local displacements using methods as described in Section 3.3. Finally propagate the displacement to the neighboring points in the current time slice using a regularization method as proposed in [ACP02, PMG*05]. Since we work on point cloud data with no connectivity, neighborhood is defined on the basis of Euclidean proximity.

Model	# scans	# points/scan (in 1000s)	Time (mins)
bunny (simulated)	314	33.8	13
bee	2,200	20.7	51
coati	2,200	28.1	71
teapot (rigid)	2,200	27.2	68
skeleton (simulated)	100	55.9	11
hand	100	40.1	17

Table 1: *Performance.* Timing in minutes for the different examples on a 2.4GHz Athlon Dual Core with 2GB RAM.

Figure 9 shows result of application to this process over a sequence of 100 scans of a deforming hand. Model consolidation is achieved by transferring data from other time slices to fill in regions of missing data. Deformable registration when applied to the teapot sequence, produces much cleaner registration at the handle (Figure 6).

Articulated Bodies. In case of models with strongly articulated components, it is possible to do better. Instead of smoothly propagating the displacement fields to neighbors, we first cluster the motion parameters $(\mathbf{c}, \bar{\mathbf{c}})$ estimated in Step 2 of our deformable registration algorithm. Based on the tightness of these clusters, we conclude if a model deformation was articulated. If so, such cluster centers are then used to derive a segmentation of the object similar to existing methods (see [MGP06] for details). This better preserves articulated parts of the scanned object. We demonstrate this method on a set of 100 simulated scans of a deforming skeleton grasp with simultaneous 'scanner' motion (Figure 10).

6. Conclusion

We demonstrated how to register multiple partial scans of a moving and/or deforming body so that a consistent 3D model of the entire object can be obtained. By assuming a fast scanner that generates a dense set of object frames, we can bypass traditional multi-view registration methods

that require the establishment of correspondences between frames. Instead, we conceptually combine all the frames into a single space-time surface, and using local kinematic properties of this space-time surface to solve for the rigid motions aligning the frames. We also extend our basic formulation for rigidly moving objects to the more challenging cases of articulated or fully deformable motion. Experimental results show the robustness and versatility of our method. In future, we plan to extend these methods to scans captured with significant non-uniform sampling (spatial or temporal). Our local tetrahedralization is the first step in this direction. Such technologies for acquiring deforming and moving shapes present a wealth of new opportunities and challenges for the area of geometry processing.

Acknowledgements. This work is supported by Austrian Science Fund grants S9206-N12 and P18865-N13, NSF grant ITR 0205671, NIH grant GM-072970, DARPA grant 32905, the Agilent Corporation, and a Neumaier fellowship. We sincerely thank Andreas Schilling for originally suggesting the idea of iterative normal refinement, and Thibaut Weise from ETH Zurich for providing the bee, coati, and teapot data sequences.

References

[ACP02] ALLEN B., CURLESS B., POPOVIC Z.: Articulated body deformation from range scan data. In *ACM SIGGRAPH* (2002), pp. 612–619.

[ASK*05] ANGUELOV D., SRINIVASAN P., KOLLER D., THRUN S., ET AL.: Scape: shape completion and animation of people. *ACM SIGGRAPH 24*, 3 (2005), 408–416.

[BB95] BEAUCHEMIN S., BARRON J.: The computation of optical-flow. *ACM Comput. Surv. 27*, 3 (1995), 433–467.

[BM92] BESL P. J., MCKAY N. D.: A method for registration of 3-d shapes. In *PAMI* (1992), vol. 14, pp. 239–256.

[BR90] BOTTEMA O., ROTH B.: *Theoretical kinematics*. Dover Publication, 1990.

[BR02] BERNADINO F., RUSHMEIER H.: The 3D model acquisition pipeline. *Compute Graphics Forum 21*, 2 (2002).

[BR07] BROWN B., RUSINKIEWICZ S.: Global non-rigid alignment of 3d scans. *ACM SIGGRAPH 26*, 3 (2007).

[BSGL96] BERGEVIN R., SOUCY M., GAGNON H., LAURENDEAU D.: Towards a general multiview registration technique. In *PAMI* (1996), pp. 540–547.

[CM92] CHEN Y., MEDIONI G. G.: Object modelling by registration of multiple range images. In *Image and Vis. Compt.* (1992), vol. 10, pp. 145–155.

[DRR03] DAVIS J., RAMAMOORTHI R., RUSINKIEWICZ S.: Spacetime stereo: A unifying framework for depth from triangulation. In *CVPR* (2003), pp. 359–366.

[ELF97] EGGERT D. W., LORUSSO A., FISHER R. B.: Estimating 3-d rigid body transformations: a comparison of four major algorithms. *Mach. Vision Appl. 9*, 5-6 (1997), 272–290.

[Fau90] FAUGERAS O.: On the motion of 3d curves and its relationship to optical flow. In *ECCV* (1990), pp. 107–117.

[FB05] FONG P., BURON F.: Sensing deforming and moving objects with commercial off the shelf hardware. In *CVPR* (2005), vol. 3, pp. 101–101.

[GMGP05] GELFAND N., MITRA N. J., GUIBAS L. J., POTTMANN H.: Robust global registration. In *Proc. Symp. Geometry Processing* (2005), pp. 197–206.

[HH03] HUBER D., HEBERT M.: Fully automatic reg. of multiple 3d data sets. *Image and Vis. Compt. 21*, 7 (2003), 637–650.

[HTB03] HÄHNEL D., THRUN S., BURGARD W.: An extension of the ICP algorithm for modeling nonrigid objects with mobile robots. In *IJCAI* (2003), pp. 915–920.

[IK66] ISAACSON E., KELLER H. B.: *Analysis of Numerical Methods*. Wiley, New York, 1966.

[KGG03] KONINCKX T. P., GRIESSER A., GOOL L. V.: Real-time range scanning of deformable surfaces by adaptively coded structured light. In *Proc. 3DIM* (2003), pp. 293–300.

[KLMV05] KRISHNAN S., LEE P., MOORE J., VENKATASUBRAMANIAN S.: Global reg. of multiple 3D point sets via opt.-on-a-manifold. In *In Proc. Sym. Geo. Proc.* (2005), pp. 187–196.

[LG05] LI X., GUSKOV I.: Multiscale features for approximate alignment of point-based surfaces. In *Proc. Symp. Geometry Processing* (2005), pp. 217–226.

[LPC*00] LEVOY M., PULLI K., CURLESS B., RUSINKIEWICZ S., ET AL.: The Digital Michelangelo Project: 3-D scanning of large statues. *ACM SIGGRAPH* (2000), 131–144.

[MA97] MOUNT D., ARYA S.: Ann: A library for approximate nearest neighbor searching, 1997.

[MGP06] MITRA N. J., GUIBAS L., PAULY M.: Partial and approximate symmetry detection for 3d geometry. In *ACM SIGGRAPH* (2006), vol. 25, pp. 560–568.

[MGPG04] MITRA N. J., GELFAND N., POTTMANN H., GUIBAS L.: Registration of pcd from a geometric opt. perspective. In *Proc. Symp. Geometry Processing* (2004), pp. 23–31.

[MNG04] MITRA N. J., NGUYEN A., GUIBAS L.: Estimating surface normals in noisy point cloud data. In *IJCGA* (2004), vol. 14, pp. 261–276.

[PGK02] PAULY M., GROSS M., KOBBELT L.: Efficient simpl. of point-sampled surfaces. In *IEEE Vis.* (2002), pp. 163–170.

[PMG*05] PAULY M., MITRA N. J., GIESEN J., GROSS M., GUIBAS L.: Example-based 3d scan completion. In *Proc. Symp. Geometry Processing* (2005), pp. 23–32.

[Pul99] PULLI K.: Multiview registration for large data sets. In *Proc. 3DIM* (1999), pp. 160–168.

[PW01] POTTMANN H., WALLNER J.: *Computational Line Geometry*. Springer, Heidelberg, 2001.

[RHHL02] RUSINKIEWICZ S., HALL-HOLT O., LEVOY M.: Real-time 3D model acquisition. *ACM SIGGRAPH 21*, 3 (2002), 438–446.

[RL01] RUSINKIEWICZ S., LEVOY M.: Efficient variants of the ICP algorithm. In *Proc. 3DIM* (2001), pp. 145–152.

[SLW02] SHARP G. C., LEE S. W., WEHE D. K.: Multiview registration of 3D scenes by minimizing error between coordinate frames. In *ECCV* (2002), pp. 587–597.

[Tur92] TURK G.: Re-tiling polygonal surfaces. *ACM SIGGRAPH 26*, 2 (1992), 55–64.

[WLG07] WEISE T., LEIBE B., GOOL L. V.: Fast 3d scanning with automatic motion compensation. In *CVPR* (2007).

[ZCS] ZHANG L., CURLESS B., SEITZ S.: Spacetime stereo: Shape recovery for dynamic scenes. In *CVPR*, pp. 367–374.