Towards understanding time varying triangle meshes

Jan Dvořák[0000−0003−4569−1151], Petr Vaněček[0000−0002−1858−2411], and Libor Váša[0000−0002−0213−3769]

Department of Computer Science and Engineering, University of West Bohemia, Pilsen, Czech Republic {jdvorak,pvanecek,lvasa}@kiv.zcu.cz

Abstract. Time varying meshes are more popular than ever as a representation of deforming shapes, in particular for their versatility and inherent ability to capture both true and spurious topology changes. In contrast with dynamic meshes, however, they do not capture the temporal correspondence, which (among other problems) leads to very high storage and processing costs. Unfortunately, establishing temporal correspondence of surfaces is difficult, because it is generally not bijective: even when the full visible surface is captured in each frame, some parts of the surface may be missing in some frames due to self-contact. We observe that, in contrast with the inherent absence of bijectivity in surface correspondence, volume correspondence is bijective in a wide class of possible input data. We demonstrate that using a proper intitialization and objective function, it is possible to track the volume, even when considering only a pair of subsequent frames at the time. Currently, the process is rather slow, but the results are promising and may lead to a new level of understanding and new algorithms for processing of time varying meshes, including compression, editing, texturing and others.

Keywords: Time varying mesh · model · animation · tracking · analysis · surface.

1 Introduction

Time Varying Meshes (TVMs) are appearing more commonly in recent years, especially because of the improved methodology (hardware and processing) used for acquiring 3D surface data at the required framerate. Photogrammetry in particular, supported by depth sensing technologies, such as time-of-flight cameras and pattern projection scanning, have enabled capturing deforming shapes, such as articulated human faces or whole bodies, with relative ease. Most of these approaches rely on a tight synchronization of the scanning devices, which allows processing data at each time instant separately, making the TVM a natural output format of the scanning process.

TVMs find application in a variety of fields, such as movie/gaming insdustry, visualization, telepresence, live sport broadcasting and others, which take advantage of the versatility of this representation. On the other hand, their application

is limited by the large size of data needed for the representation: a common sequence lasting about a minute with 100k vertices in each frame takes on the order of gigabytes to store/transmit. This is mainly because each frame carries its own connectivity, which in many cases requires a portion of the overall bit budget that is comparable with the geometry. In contrast with dynamic meshes, which share the common connectivity, this is a major disadvantage, topped by the general difficulty of exploiting the temporal coherence of the data exhibited by the TVM representation.

Another hindrance inherent to TVMs is the missing temporal correspondence. This makes certain common tasks, such as attaching an artificial prop to the 3D model, difficult. Similarly, it is difficult to exploit the temporal coherence of the surface albedo for efficient texturing using a shared map and image, since the UV unwrapping (and thus the texture itself) must be different in each frame.

Extracting the temporal correspondence would make a great step towards better understanding of the captured surface deformation. Typically, captured performances frequently involve limbs being connected or merged with each other or with the torso in the reconstruction. While human observers easily identify such occurrences and distinguish them from true merging and/or deformation, for automatic processing algorithms such distinction is difficult and leads to problems when establishing the temporal correspondence.

The main issue is that surface correspondence is inherently not bijective: in some (most) frames, certain parts of the surface are missing, even when the surface is captured from all possible viewpoints, due to the self-contact. Typically, parts of the surface may disappear for several frames and then eventually reappear. It is even possible, that the whole of the surface is not completely visible in any of the input frames.

Ideally, we would like to track the whole surface and add the invisible parts that are in contact to the frames where they are not visible. Such approach could allow for consistent, bijective correspondence tracking. Unfortunately, this is a difficult problem, which could be visualized in a simplified 2D case of tracking silhouettes, as shown in Fig. 1. The data in this case can be interpreted as a 3D (2D + time) object, and the task translates to cutting the shape along certain ridges in order to complete the silhouettes. Formulating the criteria for the cuts seems difficult and such process is necessarily prone to errors.

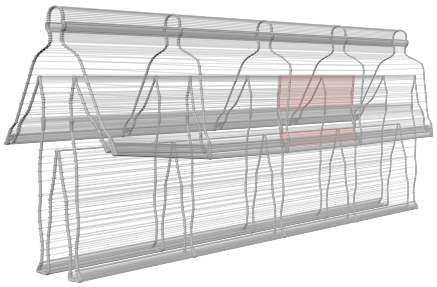

Fig. 1. A Sequence of 2D silhouettes interpreted as a 3D volume. Tracking the surface translates to cutting the 3D object at the red locations.

Our main observation is that there is a different, orthogonal approach to the problem, which eliminates much of the difficulty. Rather than tracking the surface, we propose to track the volume of the captured objects. In many practical scenarios, the overall volume changes negligibly, and it is possible to track it bijectively over the length of the sequence. Splitting the volume into finite size elements, it is much easier to formulate the criteria/priors for correct tracking: the whole of the volume in each frame should be covered, and the volume elements should move coherently. With volume correspondence, it is then much easier to establish surface correspondence and eventually add the missing parts of surface in any frame.

Our main contribution is a working volume tracking pipeline, verified on data from multiple sources. The key ingredients are:

1. choice of appropriate representation
2. proper tracking initialization
3. formulation of objective function/energy

2 Related work

One could track the evolving surface by propagating certain canonical frame (e.g. first frame) using a non-rigid alignment method. Such methods are usually based on minimizing an energy which evaluates the quality of the alignment. However, only the methods that do not require the correspondences information as an input can be considered, since this information is not known a priori. Myronenko et. al. [22,23] proposed to model the unknown correspondences using a Gaussian mixture model, enforced the movement of points to be spatially smooth and used the expectation-maximization algorithm to optimise the energy. Li et. al. [18] also treat the correspondences as probability, however, they model the movement of points using deformation graphs. In their subsequent work [2] this method was utilized to track evolving surfaces. Yoshiyasu et. al. [31] constrain the underlying deformation to be as-conformal-as-possible, i.e. preserving angles. Such property preserves the structure of the deformed mesh. Utilizing the volume the surface encloses was already considered in the past. Huang et. al. [12] proposed to use centroidal voronoi tessellations with connection to signed distance function to model volumetric features used in the alignment process. Tracking of surfaces based on non-rigid alignment has, however, quite a few limitations. Such tracking process unfortunately highly depends on the selection of the propagated frame and might suffer from the error accumulation.

Compresssion of time-varying geometry, i.e. sequence of triangle meshes or point clouds representing a surface with dynamic topology is a closely related field of research in the sense that its main goal is to find a reduced representation of the data. Quite a few methods utilizing the temporal coherence of the data have been already proposed. The most common approach is to store the surface in a spatial data structure (e.g. grid or octree) and exploit the coherence of occupancy of such structures between frames instead of the coherence of surfaces

([7,8,10,16,21,26]). Inspired by video compression methods [5,9,17,20,24,30] use motion compensation, where parts of the surface are predicted by parts of the previous frame. However, such predictions are, in general, not correspondences between frames. Since the video compression is a mature field of research, it is also utilized in so-called geometry-video coding [11, 14, 25, 29], where the geometry of the surface is mapped to a video, which is then compressed using the spatio-temporal coherence in the image domain. The temporal coherence of the surfaces themselves is exploited only in few methods. Yang et. al. proposed to match the decimated frames [15]. Another approach is to utilize skeletal information [6,19]. However, both approaches can be used only for data with constant genus.

3 Algorithm overview

We assume that the input data take the form of TVM: a sequence of triangle meshes captured at certain framerate, usually 25-30fps. Each frame consists of a set of vertex positions representing the geometry, and a set of integer triplets, representing the triangles forming the connectivity. While there is a certain inter-frame coherence of the geometry, since the subsequent frames represent similar deformations of the same shape, there is no coherence in the connectivity: it is assumed to be completely independent in each frame. For simplicity, we also assume that each input mesh is complete and watertight. Later we will discuss how this assumption can be lifted without modifying much of the proposed pipeline.

We aim at building a compact data structure that captures the temporal development of the shape represented by the input TVM. The structure should provide some form of temporal correspondence, which enables deeper insight into the semantics of the shape, as well as more efficient processing of the input data.

Our proposal is to represent the shape by a fixed number of points (denoted *centers*), each representing a small volume surrounding it. The locations of the centers will vary in time, their positions will be denoted $\mathbf{c}_i^f$, representing the 3D position of the i-th center in frame f.

In contrast with the difficulty of specifying conditions for proper surface data structure, formulating the conditions for the volume elements is comparatively easy. In particular, we wish the centers to

- cover all parts of the input object in each frame,
- be distributed evenly over the volume of the objects in each frame and
- move consistently between frames, i.e. nearby centers should move in similar direction.

Such representation can be used in many applications. It allows reconstructing a surface with naturally changing genus (as described in section 10). It can be used to track the surface and add parts missing due to self-contact. Note that

the data structure may also complement the original input instead of replacing it: in particular, it can serve as a reference for compression purposes.

When determining the model parameters (center positions in each frame), we will proceed frame by frame, considering only a pair of subsequent frames at the time. On one hand, this restriction means that the algorithm is not exploiting all the information that is available. On the other hand, such approach eventually allows for online processing of the inputs, given that the size of the data and processing power allows for real-time computations.

First, we will distribute the centers uniformly over each input shape, as will be detailed in section 4. This ensures that every part of every frame will be covered by the model.

Next, we proceed from a previous frame to the next. We determine the likely correspondence using the Kuhn-Munkers ("hungarian") method (section 5), and then we optimize an objective function described in sections 6, 7 and 8, which ensures a smooth, coherent inter-frame transition while preserving the sampling uniformity.

4 Uniform object sampling

We will be sampling the volume occupied by the object in each frame, denoted V_f, interpreted as an infinite set of points. A constant number of samples (centers) will be placed in each frame. In our experiments, 1000 centers worked well as a compromise between representation accuracy and processing time. If more centers are needed, it is easier to add them in post-processing, after the initial batch has been tracked.

We aim at achieving a uniform distribution of the centers. As with many similar problems, formulating the objective precisely goes a long way towards finding the solution. In our case, a uniform distribution can be identified by looking at the volumes that surround each center in certain sense. We may define a cell associated to i-th center as the set of all points $\mathbf{p} \in V_f$, such that $\|\mathbf{p} - \mathbf{c}_i\| \leq \|\mathbf{p} - \mathbf{c}_j\|$ for every j. Such structure is very similar to the Voronoi cell of i-th center, with the only difference that it is limited to the volume V_f. The sampling uniformity can be directly linked to the standard deviation of the cell sizes.

The objective can be thus reformulated as minimizing the spread of cell sizes. It is well known (and easy to see) that such objective can be achieved using the Lloyd iteration algorithm, where in each iteration the centers are shifted towards the center of mass of their associated voronoi cell, resulting in the so-called Centroidal Voronoi Tesselation (CVT). In order to stop the centers from diverging to infinity, the center of mass must be computed with respect to a certain weight function, in our case it is the indicator function of the volume V_f.

Despite the similarity to Voronoi cells, finding the center of mass of the cells associated with centers is not easy, since they are more general, potentially non-convex and even non-continuous (see Fig. 2). Therefore we will opt for computing the centers of mass quickly at the cost of certain approximation error.

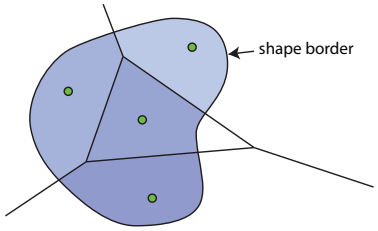

Fig. 2. Cells (shades of blue) associated with centers (green) can be non-convex and even non-continuous, but they are always finite, which contrasts the Voronoi cells, which are always convex, but often infinite.

The approach we choose is based on sampling the indicator function using a regular volume grid. The indicator function that determines for each voxel whether or not it lies within the volume occupied by the object is evaluated using the ray-shooting technique, essentially evaluating the number of intersections with the surface along a certain ray. Choosing axis aligned rays allows reusing the previously computed intersection points for evaluating a whole column of the regular sampling grid. Note that this is only possible for watertight models. A more general approach could use the generalized winding number [13] instead of the indicator function, possibly generalizing the algorithm to incomplete models, however, at this point, we have not experimented with this possibility.

Centers are initialized randomly into cells of the regular volume grid where the indicator function is positive. Next, we approximate the center of mass of each center associated cell by averaging the positions those voxel centroids, that have positive indicator function and the given center as its closest center. This is done by a single pass over all voxels, searching for the nearest center using a KD-tree. Finally, all centers are shifted to their respective associated centers of mass.

As a result, we obtain a uniform sampling of each frame. Note that the result strongly depends on the intialization ([3]), because the objective function generally exhibits a broad basin of low values, with many shallow local minima. We will exploit this property in the subsequent steps by regularizing the solution using the smoothness term.

First, however, we need to initialize the inter-frame correspondences of centers, which we detail in the next section.

5 Initial correspondence estimation

The task at hand can be formulated as an optimal transport problem, with certain cost function. We wish to find a mapping between the centers of the previous (source) and current (target) frame, such that (i) corresponding points are as close together as possible, (ii) each source point is associated to a unique

target point, or an affine combination thereof, and (iii) each target point is associated to a unique source point, or an affine combination thereof.

Apart from spatial proximity, captured by simple length squared cost function, the transport cost should also reflect whether the trajectory connecting the source and target points is located within the object, or passes outside, since the latter often indicates that the correspondence involves center points that belong to separate components or topologically distant locations (see Fig. 3). On the other hand, in case of fast movement, even correct correspondences sometimes pass "outside" of the objects, and therefore such correspondences cannot be ignored by the algorithm completely. Having the input sampled, it is relatively easy to sample every possible trajectory and adjust the cost, so that segments outside of the object are penalized: in our implementation, the penalization is done by multiplying the proportional part of the cost outside of the object by 10, which is a constant that worked well in all our experiments.

The optimal transport task at hand can be solved using the Sinkhorn iterations algorithm [4]. Each source center is associated to a weighted combination of target points, which can be averaged using their weights to obtain a unique corresponding center in the target frame. This approach has the advantage of natural smoothing of the correspondences.

In our experiments, however, we have encountered difficulty with setting the λ parameter of the Sinkhorn iterations algorithm. In essence, the parameter controls the "reach" of each center. When set too low, the algorithm does not converge to a proper solution of the problem at hand (the mapping weights were not affine), while setting it too high leads to a too poor approximation of the solution, where each source center is associated with many target centers and averaging them leads to shrinkage of the centers in the target frame. Unfortunately, in many cases it is impossible to reach a compromise value that eliminates both these effects.

Since the number of centers is constant in every frame, the problem can also be formulated as a special case of the optimal transport: the optimal assignment, where each source center is associated with exactly one target center. An optimal solution can be found efficiently using the Kuhn-Munkers ("hungarian") method. However, in contrast with the Sinkhorn iterations, the result can be a very uneven result (see Fig. 4), because of the lack of weighting and the random character of sampling of each frame. This causes problems in the following steps, because such state represents a local minimum of the objective function we are going to minimize. Fortunately, this issue can be solved using a technique described next.

6 Smoothness energy, smoothing

One of our objectives is achieving a smooth movement of centers, i.e. that nearby centers move in a similar direction. It is important to note that we do not wish to penalize the amount of movement itself: it follows from the nature of the input data that there is movement present in the sequence.

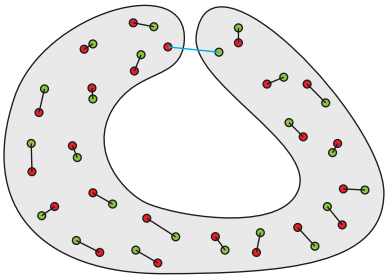

Fig. 3. Red and green dots represent different samplings of the same gray domain. The blue correspondence lies partially outside of the domain, and thus should be penalized.

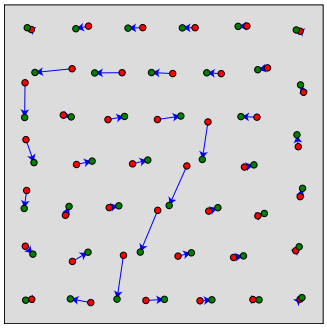

Fig. 4. Green and red dots represent stationary points of the Lloyd algorithm over the same square domain. The blue lines represent the correspondences as found by the Kuhn-Munkers method, minimizing the sum of squared distances.

The model can be interpreted as a vector field $\mathbf{v}$ defined over the domain V_f of the source frame, sampled as $\mathbf{v}(\hat{\mathbf{c}}_i) = \mathbf{c}_i - \hat{\mathbf{c}}_i = \mathbf{v}_i$, where $\hat{\mathbf{c}}_i$ is the position of the i-th center in previous frame. The object of interest is the smoothness of this vector field, which can be captured by its Laplacian $\Delta\mathbf{v}$. Smooth (harmonic) vector fields exhibit zero length Laplacian, and the sum of the squared lengths of $\Delta\mathbf{v}$ at uniform sample points can be interpreted as the amount of deviation from smoothness.

In our case, the vector field is sampled irregularly, which makes evaluating its Laplacian harder, but not impossible. In particular, the discrete Laplacian proposed by Belkin [1] can be used. This allows us to express an energy E_s that captures the smoothness of the vector field as follows:

$$E_s = \sum_{\hat{\mathbf{c}}_i \in C} \|\Delta\mathbf{v}(\hat{\mathbf{c}}_i)\|^2 = \frac{1}{|C|} \sum_{\hat{\mathbf{c}}_i \in C} \| \sum_{\hat{\mathbf{c}}_j \in C} H^t(\hat{\mathbf{c}}_i, \hat{\mathbf{c}}_j)(\mathbf{v}_j - \mathbf{v}_i)\|^2, \tag{1}$$

where $H^t(\mathbf{x}, \mathbf{y}) = \frac{1}{(4\pi t)^{\frac{5}{2}}} e^{-\frac{\|\mathbf{x}-\mathbf{y}\|^2}{4t}}$ is a Gaussian kernel with parameter t. Gradient of such energy consists of following partial derivatives:

$$\frac{\partial E_s}{\partial \mathbf{v}_k} = \frac{\partial E_s}{\partial \mathbf{c}_k} = \frac{8}{|C|^2} \sum_{\hat{\mathbf{c}}_i \in C} (H^t(\hat{\mathbf{c}}_k, \hat{\mathbf{c}}_j))^2 (\mathbf{v}_k - \mathbf{v}_j) \tag{2}$$

$$= \frac{8}{|C| \cdot (8\pi t)^{\frac{5}{2}}} \frac{1}{|C|} \sum_{\hat{\mathbf{c}}_j \in C} H^{\frac{t}{2}}(\hat{\mathbf{c}}_k, \hat{\mathbf{c}}_j)(\mathbf{v}_k - \mathbf{v}_j) \tag{3}$$

$$= -\frac{8}{|C| \cdot (8\pi t)^{\frac{5}{2}}} \hat{\Delta}\mathbf{v}(\hat{\mathbf{c}}_k), \tag{4}$$

where $\hat{\Delta}$ is a discrete Laplacian defined with half the kernel width of Δ.

7 Sampling energy

Apart from smoothness, we also need to preserve the uniform density of sampling. As discussed before, this property can be quantified as

$$E_u = \frac{1}{2} \sum_{\mathbf{c}_i \in C} \|\mathbf{m}_i - \mathbf{c}_i\|^2, \tag{5}$$

where $\mathbf{m}_i$ is the center of mass of a cell associated with i-th center.

Treating the centers of mass as constants, the gradient consists of following partial derivatives:

$$\frac{\partial E_u}{\partial \mathbf{c}_i} = \mathbf{m}_i - \mathbf{c}_i. \tag{6}$$

The location of each center of mass $\mathbf{m}_i$ can be approximated as described in section 4.

8 Overall energy

Having the two energy terms that we wish to minimize, the tracking can be formulated as minimization of the overall energy $E = E_s + \alpha E_u$, where α is a weighting constant. Having the gradient of both terms expressed in equations 4 and 6, such minimization can be done using the standard gradient descent technique, i.e. in a series of steps, the centers are shifted in the direction of the sum of the two gradients. The procedure is terminated after a set maximum of iterations has been performed, or when the gradient is sufficiently small for each $\mathbf{c}_i$.

8.1 Pre-smoothing

Note that the initial result of the Kuhn-Munkers method naturally represents a local minimum of E_u, since it maps the centers of the source frame directly to the centers of target frame, which have been optimized for uniformity during initialization. This is somewhat unfortunate, because the gradient of E_s is often not strong enough to exit the local minimum.

This issue could be addressed by a smooth variation of the weighting parameter α, starting with a stronger influence of E_s and then gradually increasing the influence of E_u. Such approach, however, leads to problems with formulating the stopping condition, because it enforces a certain number of iterations in order to reach the final ratio of term influences.

In our experiments, we have therefore used a different empirical approach: we start with a few iterations (50 worked well with our data) that only consider E_s, which lead the optimization out of the local minimum, and then we continue

with the full energy E, using a constant value of α. This way, we can stop the iteration whenever the gradient is small enough (smaller than 0.1 mm in each component in our experiments), since the energy expression does not change during the descent.

9 Results

We have tested the proposed algorithm on data from two sources: a commercially available TVM sequence of professional quality (denoted *casual_man*), and a pair of sequences provided for academic purposes (denoted *samba* and *handstand*), consisting of mesh sequences of constant connectivity [28]. Note that even though the second dataset consists in fact of dynamic meshes rather than TVMs, we do not exploit this fact and treat the datasets as if they were TVMs. Table 1 summarizes some basic properties of the used input datasets.

Table 1. Datasets used in experiments.

Dataset	no. of frames (average)	no. of vertices	uncompressed size (.obj)[MB]
casual_man	546	36 145	4 086
samba	175	9 971	113
squat	250	10 002	159

Using a volume grid of 512 cells along the longest dimension and a stopping condition of 0.1mm in each component of the gradient descent direction, we have processed the *casual_man* dataset at 30.7s per frame on average. This framerate was achieved using a purely CPU implementation written in C#, including all the necessary steps (volume grid sampling, initial point distribution, energy minimization). We have used a common Intel Core-i7 9700 CPU. As for the other datasets, the processing time per frame was only slightly shorter, because most of the processing cost depends on the resolution of the volume grid, which was constant over all the experiments. The results are shown in Fig. 5. The means of quantifying the tracking quality naturally depends on the particular application, some of which we discuss next.

10 Applications

It is possible to interpret the resulting model as an approximate representation of the input deformable surface. A mesh can be extracted from the set of centers in each frame by extracting the iso-surface of an appropriately designed function that represents the distance from the centers, ideally blended in some way. One particular choice is using

$$f(\mathbf{x}) = -ln\Big(\sum_{\mathbf{c}_i \in C} exp\big(-k(\|\mathbf{c}_i - \mathbf{x}\| - r)\big)\Big)/k, \tag{7}$$

Fig. 5. Tracking results, showing the first, middle and last frame for each sequence. Coloring is done by the first three principal trajectory coefficients, notice the consistency of the coloring throughout the sequence.

where r is a minimum radius of sphere surrounding each center and k is a parameter influencing the blending distance. The function is sampled using a regular grid and its iso-surface can be extracted using a standard technique, such as marching cubes. The resulting triangle mesh can be compared with the input using some tessellation oblivious metric, such as the mean nearest inter-surface distance. Fig. 6 shows the resulting Hausdorff distances for the *casual_man* sequence, using 1000, 2000 and 4000 centers.

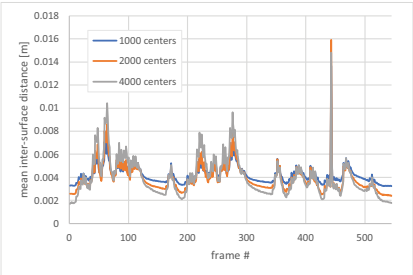

Fig. 6. Mean inter-mesh distances for the *casual_man* dataset.

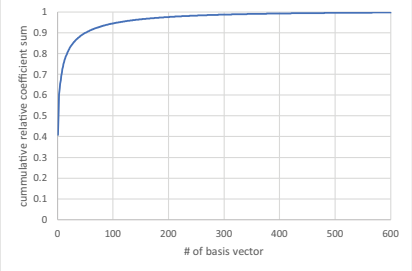

Fig. 7. Cummulative sums of coefficient magnitudes. 90% of coefficients are captured by the first 50 eigenvectors, 99% are captured by 346 eigenvectors.

We also observe that the trajectories of the centers are located in a small subspace of the full space of trajectories. A basis of the space can be found using PCA, similarly to what has been previously done for vertex trajectories in dynamic meshes [27]. A reduced basis can be used to capture most of the variance present in the data, leading to even more efficient representation. Fig. 7 shows the cumulative relative sums of coefficient magnitudes, showing that most of the variance is concentrated with the first few eigenvectors.

The model can also be used as a compressed version of the input data. Storing the complete data needed for reconstruction, i.e. 1000 trajectories, using 32-bit floats for coordinates, amounts to roughly 6MB of data for the *casual_man* sequence. Projecting onto the most important eigentrajectories, it is possible to reduce the amount data down to about 2MB without sacrificing the quality. Should a TVM be compressed down to equivalent size, each frame must be stored in no more than 3.7kB. Current state of the art compression techniques require about 16 bits per vertex, i.e. each frame must be simplified from the original 36k vertices down to 1830, which necessarily drastically degrades the visual quality.

11 Conclusions

We have described a volume tracking algorithm for analyzing Time-Varying meshes. It is based on simple energies, yet it produces feasible results, both perceptually and numerically, despite the fact that the tracking is done using the information from only a pair of subsequent shapes. As far as we know, there is currently no competing method that is able to analyze a sequence of meshes with the generality and precision provided by our method.

This result can find many applications. The resulting model can be used to analyze the deforming bodies: it captures the nature of the deformation, it allows tracking volumes even when they visually merge with others or get occluded or hidden.

In the future, we would like to work on speeding the process up, potentially using a GPU implementation of some of the time consuming steps. Additionally, we would like develop means increasing the number of tracked volumes, yielding a finer representation. We believe that with a proper initialization provided by the proposed method, tracking additional volumes should be manageable. Finally, we would like to better analyze the relations of the centers, building some kind of connectivity that captures which volumes move together and how. Having such structure could in turn help with further improving the tracking.

12 Acknowledgement

This work was supported by the project 20-02154S of the Czech Science Foundation. Jan Dvořák was also partially supported by the University specific student research project SGS-2019-016 Synthesis and Analysis of Geometric and Computing Models. The authors thank Diego Gadler from AXYZ Design, S.R.L. for providing the test data.

References

1. Belkin, M., Sun, J., Wang, Y.: Discrete laplace operator on meshed surfaces. In: Proceedings of the Twenty-Fourth Annual Symposium on Computational Geometry. p. 278–287. SCG '08, Association for Computing Machinery, New York, NY, USA (2008)

2. Bojsen-Hansen, M., Li, H., Wojtan, C.: Tracking surfaces with evolving topology. ACM Trans. Graph. **31**(4) (Jul 2012)
3. Celebi, M.E., Kingravi, H.A., Vela, P.A.: A comparative study of efficient initialization methods for the k-means clustering algorithm. Expert Syst. Appl. **40**(1), 200–210 (2013)
4. Cuturi, M.: Sinkhorn distances: Lightspeed computation of optimal transport. In: Burges, C.J.C., Bottou, L., Welling, M., Ghahramani, Z., Weinberger, K.Q. (eds.) Advances in Neural Information Processing Systems. vol. 26, pp. 2292–2300. Curran Associates, Inc. (2013)
5. de Queiroz, R.L., Chou, P.A.: Motion-compensated compression of dynamic voxelized point clouds. IEEE Transactions on Image Processing **26**(8), 3886–3895 (2017). https://doi.org/10.1109/TIP.2017.2707807
6. Doumanoglou, A., Alexiadis, D.S., Zarpalas, D., Daras, P.: Toward real-time and efficient compression of human time-varying meshes. IEEE Transactions on Circuits and Systems for Video Technology **24**(12) (2014). https://doi.org/10.1109/TCSVT.2014.2319631
7. Garcia, D.C., de Queiroz, R.L.: Context-based octree coding for point-cloud video. In: 2017 IEEE International Conference on Image Processing (ICIP). pp. 1412–1416 (2017). https://doi.org/10.1109/ICIP.2017.8296514
8. Garcia, D.C., Fonseca, T.A., Ferreira, R.U., de Queiroz, R.L.: Geometry coding for dynamic voxelized point clouds using octrees and multiple contexts. IEEE Transactions on Image Processing **29**, 313–322 (2020). https://doi.org/10.1109/TIP.2019.2931466
9. Han, S., Yamasaki, T., Aizawa, K.: Time-varying mesh compression using an extended block matching algorithm. IEEE Transactions on Circuits and Systems for Video Technology **17**(11), 1506–1518 (2007). https://doi.org/10.1109/TCSVT.2007.903810
10. Han, S., Yamasaki, T., Aizawa, K.: Geometry compression for time-varying meshes using coarse and fine levels of quantization and run-length encoding. In: 2008 15th IEEE International Conference on Image Processing. pp. 1045–1048 (2008). https://doi.org/10.1109/ICIP.2008.4711937
11. Hou, J., Chau, L., He, Y., Magnenat-Thalmann, N.: A novel compression framework for 3d time-varying meshes. In: 2014 IEEE International Symposium on Circuits and Systems (ISCAS). pp. 2161–2164 (2014). https://doi.org/10.1109/ISCAS.2014.6865596
12. Huang, C., Allain, B., Franco, J., Navab, N., Ilic, S., Boyer, E.: Volumetric 3d tracking by detection. In: 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 3862–3870 (2016). https://doi.org/10.1109/CVPR.2016.419
13. Jacobson, A., Kavan, L., Sorkine, O.: Robust inside-outside segmentation using generalized winding numbers. ACM Trans. Graph. **32**(4) (2013)
14. Jang, E.S., Preda, M., Mammou, K., Tourapis, A.M., Kim, J., Graziosi, D.B., Rhyu, S., Budagavi, M.: Video-based point-cloud-compression standard in mpeg: From evidence collection to committee draft [standards in a nutshell]. IEEE Signal Processing Magazine **36**(3), 118–123 (2019). https://doi.org/10.1109/MSP.2019.2900721
15. Jeong-Hyu Yang, Chang-Su Kim, Sang-Uk Lee: Semi-regular representation and progressive compression of 3-d dynamic mesh sequences. IEEE Transactions on Image Processing **15**(9), 2531–2544 (2006). https://doi.org/10.1109/TIP.2006.877413

16. Kammerl, J., Blodow, N., Rusu, R.B., Gedikli, S., Beetz, M., Steinbach, E.: Real-time compression of point cloud streams. In: 2012 IEEE International Conference on Robotics and Automation. pp. 778–785 (2012). https://doi.org/10.1109/ICRA.2012.6224647
17. Kathariya, B., Li, L., Li, Z., Alvarez, J.R.: Lossless dynamic point cloud geometry compression with inter compensation and traveling salesman prediction. In: 2018 Data Compression Conference. pp. 414–414 (2018). https://doi.org/10.1109/DCC.2018.00067
18. Li, H., Sumner, R.W., Pauly, M.: Global correspondence optimization for non-rigid registration of depth scans. In: Computer graphics forum. vol. 27, pp. 1421–1430. Wiley Online Library (2008)
19. Lien, J.M., Kurillo, G., Bajcsy, R.: Multi-camera tele-immersion system with real-time model driven data compression. The Visual Computer **26**(1), 3 (2010). https://doi.org/10.1007/s00371-009-0367-8
20. Mekuria, R., Blom, K., Cesar, P.: Design, implementation, and evaluation of a point cloud codec for tele-immersive video. IEEE Transactions on Circuits and Systems for Video Technology **27**(4), 828–842 (2017). https://doi.org/10.1109/TCSVT.2016.2543039
21. Milani, S., Polo, E., Limuti, S.: A transform coding strategy for dynamic point clouds. IEEE Transactions on Image Processing **29**, 8213–8225 (2020). https://doi.org/10.1109/TIP.2020.3011811
22. Myronenko, A., Song, X.: Point set registration: Coherent point drift. IEEE transactions on pattern analysis and machine intelligence **32**(12), 2262–2275 (2010)
23. Myronenko, A., Song, X., Carreira-Perpinán, M.A.: Non-rigid point set registration: Coherent point drift. In: Advances in neural information processing systems. pp. 1009–1016 (2007)
24. Santos, C.F., Lopes, F., Pinheiro, A., da Silva Cruz, L.A.: A sub-partitioning method for point cloud inter-prediction coding. In: 2018 IEEE Visual Communications and Image Processing (VCIP). pp. 1–4 (2018). https://doi.org/10.1109/VCIP.2018.8698661
25. Schwarz, S., Sheikhipour, N., Fakour Sevom, V., Hannuksela, M.M.: Video coding of dynamic 3d point cloud data. APSIPA Transactions on Signal and Information Processing **8**, e31 (2019). https://doi.org/10.1017/ATSIP.2019.24
26. Thanou, D., Chou, P.A., Frossard, P.: Graph-based compression of dynamic 3d point cloud sequences. IEEE Transactions on Image Processing **25**(4), 1765–1778 (2016). https://doi.org/10.1109/TIP.2016.2529506
27. Váša, L., Skala, V.: Coddyac: Connectivity driven dynamic mesh compression. In: 3DTV Conference Proceedings (2007)
28. Vlasic, D., Baran, I., Matusik, W., Popović, J.: Articulated mesh animation from multi-view silhouettes. ACM Trans. Graph. **27**(3), 1–9 (Aug 2008)
29. Xu, Y., Zhu, W., Xu, Y., Li, Z.: Dynamic point cloud geometry compression via patch-wise polynomial fitting. In: ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). pp. 2287–2291 (2019). https://doi.org/10.1109/ICASSP.2019.8682413
30. Yamasaki, T., Aizawa, K.: Patch-based compression for time-varying meshes. In: 2010 IEEE International Conference on Image Processing. pp. 3433–3436 (2010). https://doi.org/10.1109/ICIP.2010.5652911
31. Yoshiyasu, Y., Ma, W.C., Yoshida, E., Kanehiro, F.: As-conformal-as-possible surface registration. Computer Graphics Forum **33**(5), 257–267 (2014). https://doi.org/https://doi.org/10.1111/cgf.12451