CS231A Project Report: Real-Time Interactive Airbending

Mridul Aanjaneya
Stanford University
aanjneya@stanford.edu

Michael Lentine
Stanford University
mlentine@stanford.edu

Abstract

We propose a framework for real-time interactive physics-based simulation using state of the art motion sensing devices such as the Microsoft Kinect. The user is able to interact with the system using simple hand gestures, where each gesture has a predefined action associated with it. Our gesture recognition algorithm employs multiclass SVM's, and is both rotation invariant as well as robust to noise. To achieve real-time interactive rates, our physics simulation engine has been parallelized and draws upon recent algorithms that reduce the cost of individual steps and help take large time steps while still preserving visual fidelity. We present several qualitative and quantitative evaluation results to demonstrate the robustness of our method.

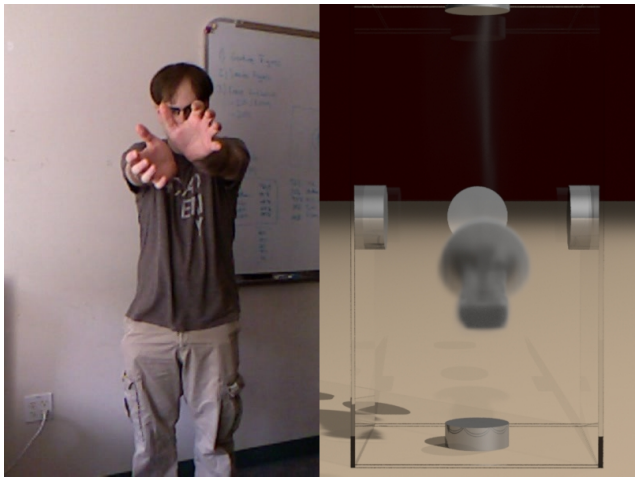

Figure 1. Gesture matching of a shooting motion and the resulting simulation.

1. Introduction

Numerical simulation has been an extremely important computational method used for understanding natural phenomena which are not directly amenable to theoretical analysis. Unfortunately, current simulation algorithms suffer from the curse of dimensionality, and the computational overhead incurred by running these algorithms is substantial. Moreover, the processes of debugging, evaluation, and verification are exacerbated in three spatial dimensions. Thus, new algorithms are required which can accelerate the speed of these intensive simulations and also allow for direct interaction. An interactive platform would not only increase a simulation programmer's understanding of a particular algorithm but also ease the process of simulating complex test cases and facilitate more thorough testing. Motivated along these lines, we propose a framework for real-time interactive physics-based simulation which expands the domain of possible inputs to hand gestures using state of the art motion sensing devices such as the Microsoft Kinect [2]. Figure 1 shows an example result from our system and Figure 2 shows some gestures it currently supports. Our framework leads to a more intuitive real-time interactive physics engine since gestures are perhaps the most natural form of interaction.

To design such an interactive framework, fast algorithms are required for both gesture recognition and physics-based simulation. While gesture recognition has been an active area of research, most available systems are constrained to recognizing two input gestures which are free from noise. A naïve approach towards gesture recognition, which we call the *norm of distances*, is to first rescale the two input gestures to be of the same length using linear interpolation and compute a similarity measure as the sum of all pairwise distances between corresponding poses, where the poses are normalized to have zero mean. Although such an approach could provide reasonable results, it has a few caveats. First, by rescaling the two gestures, the algorithm implicitly assumes that they are uniformly sampled in time, which is typically not a valid assumption. Secondly, it is not rotation invariant, since it does not align corresponding poses before computing pairwise distances. Hence, if the user performs the same gesture in two different directions, norm of distances will incorrectly label them as two distinct gestures.

For gesture recognition, we require our system to be able to recognize logically similar gestures. This requires that the underlying metric encodes the semantic meaning of an action, as opposed to just the extrinsic 3D embedding. For example, if one wants to shoot a fireball to the right and to the left, the performed action remains the same, but the

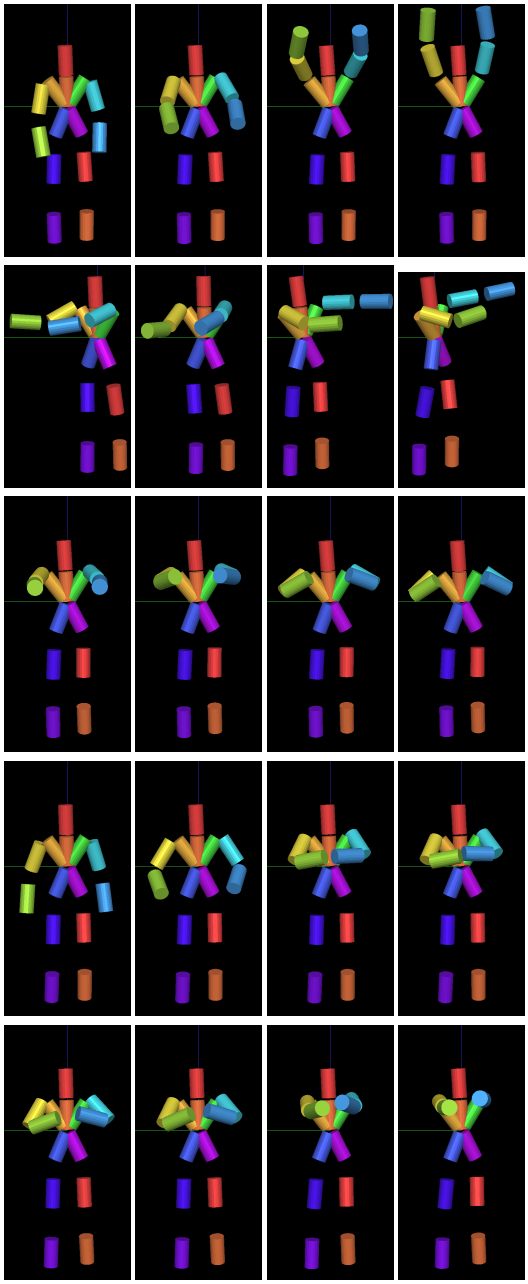

Figure 2. Key poses for some gestures recognized by our system. From top to bottom, gestures are: up, left, back, ball, shoot. The up, left and back gestures have their corresponding analogues down, right and front gestures as well. The ball gesture is for creating a ball of smoke, and the shoot gesture fires this ball.

user shoots in two different directions. To achieve this, many authors use either *dynamic timewarping* [8, 23, 24] or Hidden Markov Models [12, 39]. The latter class of methods learn a transition probability between gestures. In a real-time simulation system, there typically is no correlation between consecutive gestures, and hence, we chose the dynamic timewarping approach. An advantage of this approach is that it does not assume that the two input gestures are uniformly sampled in time. The rotation invariance of gestures is achieved by using the distance function proposed by Kovar et al. [25] which first computes an optimal alignment between input poses before computing pairwise distances between corresponding points. Since we require our framework to recognize gestures interactively, we need a candidate descriptor for each given gesture to be able to compare it against the input data. One way to compute this candidate descriptor is to pre-record an instance of the given gesture. However, such descriptors can be noisy and perform poorly when considering multiple users where input skeletons have variable dimensions. Also in a realistic setting, several users may be interacting with a particular physics simulation, such as while training new recruits in the army, or when a teacher is explaining a theoretical concept to her student. Hence, both the candidate descriptor for a given gesture as well as the recognition algorithm must take into account the variance in the input skeletons. One way to alleviate these issues is to average a number of input gestures from various users. However, manually labeling a big set of gestures can be quite tedious. Hence, we employ K-means clustering with our training data which automatically detects dominant clusters. Each cluster then represents a candidate gesture, and the descriptor is computed as the average of all the gestures within a cluster, where each gesture is rescaled to be of the same length using linear interpolation.

While dynamic timewarping performs well when gestures are fairly distinct, in a real-time scenario where motion data comes in as a continuous input stream it becomes difficult for the recognition algorithm to pick starting and ending frames for a given gesture accurately. Hence, gesture matches are often missed. For example, most users perform the up gesture followed by the down gesture, and if the recognition algorithm does not accurately locate the boundary, one of them will be missed. One way to resolve this issue would be to choose some threshold value δ, and if the similarity measure between the closest gesture matching the input stream is less than δ then the corresponding action is peformed and the input stream is cleared out, otherwise the closest gesture is ignored. In practice, since users have their own idiosyncracies while performing gestures, we found it difficult to pick a reasonable threshold value. If the threshold was too high then the recognition algorithm would not match gestures at all, and if the threshold was too low, every gesture would be a match (mostly incorrect) and the input stream would be cleared out with a very high frequency. Instead, we employ a multiclass Support Vector Machine (SVM) based method for gesture recognition which can robustly identify corresponding gestures even when input data has a large variance. Although this approach works well for real-time recognition with noisy motion data, we do note that it requires a significantly larger training set than that

required by dynamic timewarping for gesture recognition.

For designing an interactive physics engine, we considered smoke simulations. Methods for simulating smoke are either grid-based (Eulerian) [17, 37, 16] or particle-based (Lagrangian) [32, 14]. Smoke solvers typically have two main phases: *advection*, which moves around smoke density from one part of the domain to another, and *projection*, which makes the underlying velocity field divergence-free. Typically when designing a real-time smoke solver, the computational limits force the resulting simulation to be visually unappealing because of low resolutions. To alleviate this issue, we designed an interactive grid-based smoke solver which draws upon recent algorithms that speed up both the projection [27] and the advection phase [26] of a typical smoke simulation by orders of magnitude. Moreover, to achieve high levels of realism at real-time interactive rates, we parallelized our implementation using threading. Using a method similar to Yoon et al. [42] we added details for increased visual fidelity.

Apart from its obvious advantage in facilitating software development, we believe our framework can be useful for educational purposes, since virtual classroom environments, facilitated by the infrastructure of the Internet, can undoubtedly provide new and unique learning experiences. Moreover, special effects industries can benefit from our system as well, since effects and animation artists often try to get the correct look in a scene by 'tweaking' physical simulations of cloth, water, fire, and more. The proposed framework can also be utilized for national defense purposes. The United States Army has developed a video game called America's Army [1], which aims to provide information and a realistic virtual setting to potential new recruits, and as such can clearly benefit from a real-time interactive high quality physics engine to increase visual realism.

The rest of the paper is organized as follows: Section 2 discusses related work. Section 3 discusses our approach towards gesture recognition and real-time physics simulation. We present our experimental results and evaluations in Section 4. We conclude in Section 5 discussing avenues for future work.

2. Related Work

Our work lies in the intersection of Computer Vision and Computer Graphics, and hence, draws upon existing work from both these communities. Gesture recognition has been an active area of research in computer vision for more than a decade, and we briefly review the two most related approaches to our problem. The first approach treats a gesture as a temporal sequence of poses. Hence, the problem of gesture recognition reduces to the problem of string matching where each element of the string is a pose descriptor [25, 23, 24]. The second approach encodes an entire gesture which is achieved by treating a gesture as a motion curve in a high-dimensional space and using dimensionality reduction techniques [33, 4] such as Principal Component Analysis (PCA) or Multidimensional Scaling (MDS). Individual gestures are then matched using state of the art curve matching algorithms [41, 19].

Both the above approaches have their advantages and disadvantages. For example, dynamic timewarping methods explicitly encode more geometric information since they use pose descriptors, while removing noise from such representations becomes more challenging. In contrast, motion curve approaches handle noise quite easily since it manifests itself as high frequency oscillations along the curve which can be removed by using fourier analysis techniques. However, a lot of geometric information often gets lost when dimensionality reduction techniques are used to embed these curves in lower dimensions.

There has been a plethora of work on simulating smoke, and more generally incompressible fluids, in the graphics community. As mentioned previously, methods for simulating smoke are either grid-based (Eulerian) or particle-based (Lagrangian). Eulerian methods, being limited by the size of the grid, often add noise to produce details [38, 31, 7, 22, 34]. These techniques, although successful at adding details, can still be somewhat expensive and are nonphysical. Higher order accurate methods have been proposed for improving the baseline simulation on the existing grid [15, 35]. However, they are significantly more expensive than traditional fluid simulation and are limited by the Nyquist frequency of the grid. To increase the grid resolution with minimal computational overhead, adaptive grid techniques were introduced [5, 28]. Lagrangian methods, on the other hand, are not limited by the resolution of the grid. However, these approaches do not store the connectivity of the surface and require additional computation to keep track of the surface and to re-mesh. There has also been work on combining particle and grid-based approaches [36, 18, 30], using reduced order models [40] and creating higher resolution results from lower resolution simulations [29, 21, 42].

3. Technical Approach

Using Microsoft's Kinect is central to our problem. We used the openni device drivers to read data, which has color and depth information, and the NITE skeleton tracker to obtain a set of joint angles, as shown in Figure 3 (Left). Once the joint angles were available, we converted them into an appropriate skeleton which is better suited as input data for our gesture recognition algorithm. To compute the skeleton, we constructed a set of cylinders for each bone by calculating the averages between positions of each joint and then using the relative joint positions to compute bone orientations. We use OpenGL to display the results of our skeleton tracking algorithm (see Figure 3 (Right)). We collected training data by using our skeleton tracking algorithm and

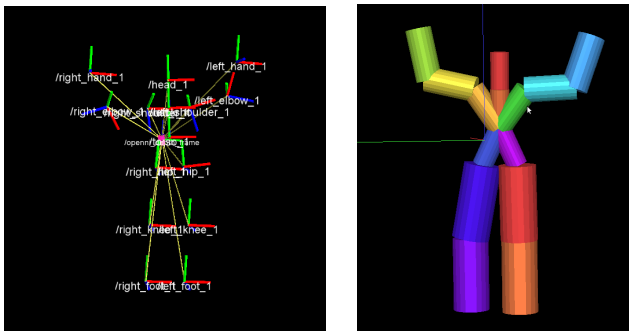

Figure 3. (Left) capture of joint angles using Kinect data and (Right) a skeleton created from cylinders.

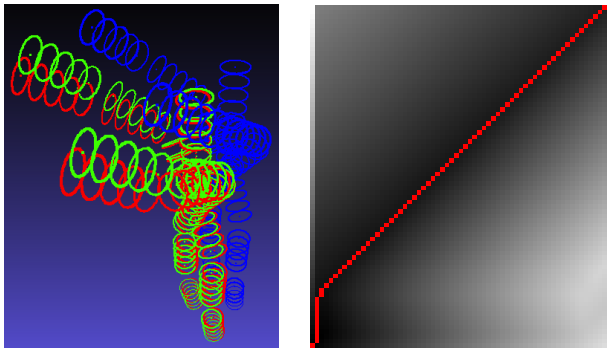

Figure 4. (Left) Original point clouds are shown in red and blue, green is the result of aligning blue to red. (Right) Comparing time alignments (red). Darker pixels show smaller frame distances.

manually cropping frames containing relevant gestures.

We adopted two different approaches to gesture recognition. Our first approach computes a similarity metric between two gestures, while the second approach performs classification among the set of candidate gestures. In the following sections we explain both these approaches.

3.1. Distance-based Gesture Recognition

For recognition, each gesture needs to be described by a set of parameters. We adopted a data-driven approach for computing these parameters by clustering our training data. We used K-means clustering for this purpose, where K is the number of gestures we wish to recognize. The parameters for each candidate gesture were computed as the average of all the gestures within a cluster, where each gesture was normalized to be of the same length through linear interpolation. Since the input gestures are inherently noisy, we applied smoothing before clustering by convolving each gesture with a Gaussian filter. This was done for each joint independently in a dimension by dimension manner.

When comparing similarity between two poses, an optimal alignment first needs to be computed to ensure direction and viewpoint invariance. For this purpose, we used the distance function introduced by Kovar et al. [25]. Each pose is represented as a point cloud derived from the input skeleton. Distance between two poses is computed as the weghted sum of squared distances between corresponding points $\mathbf{p}_i$ and $\mathbf{p}'_i$ in the two point clouds $\mathcal{P}_1$ and $\mathcal{P}_2$, i.e.,

$$d(\mathcal{P}_1, \mathcal{P}_2) = \min_{\theta, x_0, y_0} \sum_i ||\mathbf{p}_i - \mathbf{T}_{\theta,x_0,y_0}\mathbf{p}'_i||^2 \qquad (1)$$

The transformation $\mathbf{T}_{\theta,x_0,y_0}$ rotates a point $\mathbf{p}$ about the Z (vertical) axis by θ degrees and then translates it by (x_0, y_0). Hence, equation (1) computes the minimal weighted sum of squared distances, given that an arbitrary rigid 2D transformation may be applied to the second point cloud. This optimization problem has a closed-form solution [25] given by

$$\theta = \arctan \frac{\sum_i w_i(x_i y'_i - x'_i y_i) - \frac{1}{\sum_i w_i}(\bar{x}\bar{y}' - \bar{x}'\bar{y})}{\sum_i w_i(x_i x'_i + y_i y'_i) - \frac{1}{\sum_i w_i}(\bar{x}\bar{x}' + \bar{y}\bar{y}')}$$

$$x_0 = \frac{1}{\sum_i w_i}(\bar{x} - \bar{x}'\cos\theta - \bar{y}'\sin\theta)$$

$$y_0 = \frac{1}{\sum_i w_i}(\bar{y} + \bar{x}'\sin\theta - \bar{y}'\cos\theta)$$

where $\bar{x} = \sum_i w_i x_i$ and the other terms $\bar{x}'$, $\bar{y}'$ and $\bar{y}$ are defined similarly. Figure 4 (Left) shows a sample result of the point cloud alignment on our input skeleton. For computing the given alignment, we assumed all points to have the same weight w_i equal to 1. However, since we only consider hand gestures in our experiments, we assigned a large weight to all points on the hands.

Given a distance metric between any two poses, a measure of similarity between the two corresponding gestures can be computed by aligning them in time. Note that rescaling the two gestures to be of the same length, as done by the norm of distances approach, is not a good solution for this problem since the two gestures might have been performed with varying speeds. *Dynamic timewarping*, which is closely related to the *edit-distance* computation problem between two input strings [13], is a well-known algorithm for computing an optimal time alignment between two gestures. Several dynamic programming algorithms exist for this problem [8, 23]. If the two input gestures are of lengths m and n, we use the distance function defined in equation (1) to compute an $m \times n$ cost matrix C. Each entry $C(i, j)$ in the matrix contains the distance between frame i in the first gesture and frame j in the second gesture. Given such a cost matrix C, dynamic timewarping computes the path with minimum cost connecting the bottom left corner $(1, 1)$ and the top right corner (m, n). This is done by solving the

optimization problem

$$D(m,n) = \begin{array}{c} \min\{D(m-1,n-1)+C(m,n), \\ D(m,n-1)+\varepsilon, \\ D(m-1,n)+\varepsilon\} \end{array}$$

where the cost of a path is defined as the sum of the costs of each of its cells, ε is the cost of skipping a frame and $D(i,j)$ is the optimal cost of aligning the first i frames of the first gesture with the first j frames of the second gesture (see [8, 23] for details). Figure 4 (Right) shows the cost matrix C for two gestures of hands rising in the air performed by the same user. An optimal time alignment between the two gestures computed using the dynamic timewarping algorithm is shown in red. Note that the first few frames in the second gesture are skipped since the two gestures are of slightly different lengths.

3.2. SVM-based Gesture Classification

In the presence of a multitude of data, SVM-based methods lend themselves well as they adopt a more learning based approach. Typically SVM methods are broken up into two categories: one-vs-all and one-vs-one which have been extensively studied [6, 20]. In one-vs-all methods, n classifiers are found. Using these classifiers a new data point can be categorized by using the classifier that produces the highest score value. In contast, one-vs-one SVM methods find classifiers between every pair. When a data point is categorized, each classifier votes for the correct category for that particular classifier and the category with the largest number of votes wins. Following the recommendations presented by Hsu and Lin [20], we use one-vs-one SVM's.

We use the implementation provided in LibSVM [11]. Features vectors are defined as the set of positions and velocities of each bone in the input skeleton. The velocities are computed using a standard central difference convolution operator on the set of positions. We use C-SVC with cost equal to 10, and radial basis functions for the kernel where γ is inversely proportional to the number of data points within a single gesture in the training set. The training set we used contained the input gestures themselves as well as many combinations. This was done to find correct gestures even when other gestures were in the live stream before this gesture was performed.

3.3. Real-time Interactivity

Our method uses the framework proposed by Fedkiw et al. [16] for smoke simulation. For additional performance we use the algorithm recently proposed by Lentine et al. [27] for simulating fluids on high resolution grids by using a coarse grid for enforcing incompressibility. This allows for fast implicit updates while still maintaining the high quality results obtained with the large grid. In a typical fluid simulation, the bottleneck is the cost of enforcing

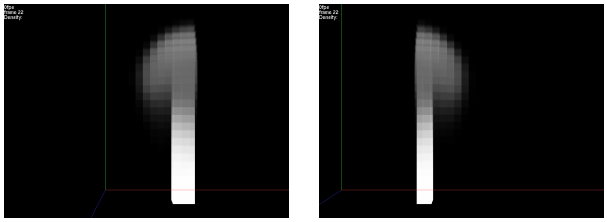

Figure 5. (Left) output of the thread associated with the left side of the domain and (Right) a thread on the right side of the domain.

incompressibility, since it involves a global Poisson solve to compute a pressure which makes the underlying velocity field divergence-free. The novelty of this method lies in significantly decreasing the cost of this global projection. Additionally, because the projection step needs to be solved at each time step, larger time steps would require fewer projection solves, speeding up the overall simulation. To achieve this, we use the algorithm recently proposed by Lentine et al. [26] which allows one time step to be taken per frame by accurately conserving mass and momentum of the fluid. Because of these techniques we can run low resolution grid-based simulations with real-time performance.

We further improve this performance and achieve significantly higher resolution simulations while maintaining the real-time constraint by using two methods. Firstly, we add parallelism to these simulations using threading by breaking down the underlying grid into a number of smaller pieces. Each thread then independently simulates it's own piece and only communicates boundary data between the pieces when needed for simulation (as shown in Figure 5). The second step is to increase the visual fidelity after the simulation is done by convolving the density field with an upsampling filter that increases visual details. We achieve this by breaking up our problem by dimension and applying the filter g on each dimension given by

$$g(x) = \begin{cases} \frac{1}{2}\left[f(\lfloor \frac{x}{2} \rfloor) \quad 0 \quad f(\lceil \frac{x}{2} \rceil)\right] & : x \text{ is odd} \\ \left[0 \quad f(\frac{x}{2}) \quad 0\right] & : x \text{ is even.} \end{cases}$$

4. Experimental Results

We tested our system on a data set of 155 gestures performed by a number of different users, where each user performed the same gesture multiple times, but not all gestures were performed by every user. The users had varying heights and each user had a different way of performing a given gesture. These variances are amply reflected in our dataset. Relevant gestures from each user were manually cropped. In what follows, we use the following terminology. Each candidate gesture computed using Gaussian filtering and K-means clustering (see Section 3.1 for details) are simply referred to as *candidate gestures*. Gestures performed by the user in real-time are referred to as *instances*. The candidate gesture corresponding to a given instance is

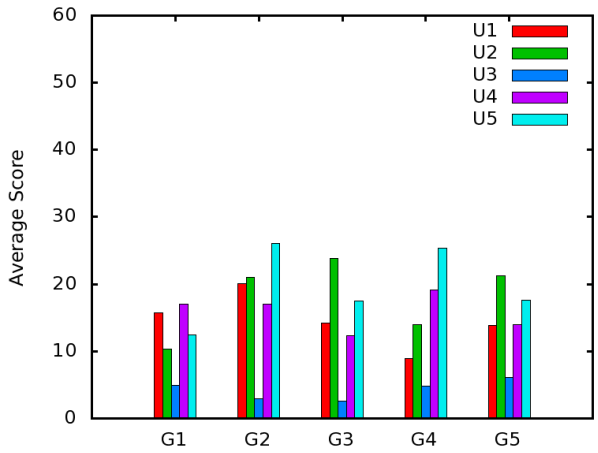

Figure 6. Matching scores for the norm of distances approach.

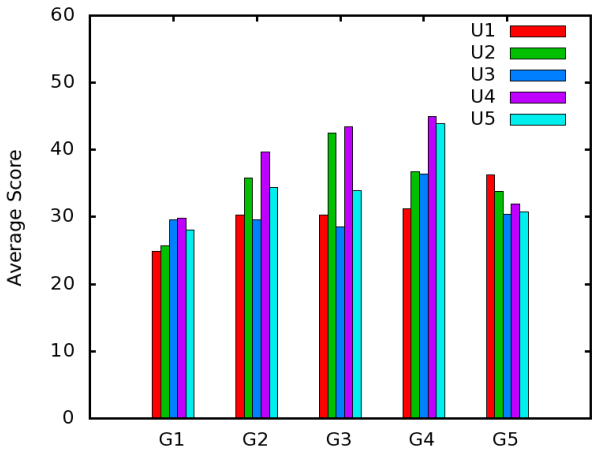

Figure 7. Difference scores for the norm of distances approach.

called a *canonical gesture*. For example, if the user performed the 'up' gesture, then the candidate gesture for the upward motion is the corresponding canonical gesture. Two instances are called *similar* if they have the same canonical gesture. Candidate gestures which do not correspond to a given instance are called *non-canonical gestures*. We used the following metrics for evaluating the performance of a given recognition algorithm:

- **Matching Score:** For each user, we compute the average similarity scores between instances and canonical gestures. For good recognition, these score values should ideally be low for all users.

- **Difference Score:** For each user, we compute the average similarity scores between similar instances and non-canonical gestures. In our dataset there are eight candidate gestures, and so each instance has seven non-canonical gestures. For good recognition, these score values should ideally be high for all users.

- **Matching Accuracy:** For each user, we compute the percentage of canonical gestures which also had the smallest similarity score when comparing an instance to the set of candidate gestures. Ideally, each instance should give the smallest similarity score to the canonical gesture among the set of candidate gestures.

For each recognition algorithm, we computed the matching accuracy in two different ways: one approach used all the testing data for training, and the other approach used 4-fold cross-validation, where the data for one user was used for testing, while those for others was used as training data. The matching accuracy for cross-validation was computed as the average of all the matching accuracies where each user was left out exactly once in the training set.

4.1. Norm of Distances

Figure 6 shows the matching scores and Figure 7 shows the difference scores for the norm of distances approach. Although this approach is naïve in the sense that it neither computes pose distances by aligning corresponding point together, nor does it optimally align the two gestures in time, it is still able to reasonably distinguish non-matching gestures. However, as can be seen, there are inconsistencies where this approach gets confused. For example, the matching score for user 5 for gestures 2 and 4 are almost the same as the difference score for user 1 for gesture 1. Hence, using this approach makes it difficult to pick a global threshold which holds for all users and serves as a separator for when an instance matches a candidate gesture. Figure 9 (red) shows the matching accuracy when all the testing data was used for training, and Figure 11 (red) shows the matching accuracy with 4-fold cross-validation.

4.2. Dynamic Timewarping

Figure 8 shows the matching scores and Figure 10 shows the difference scores for the dynamic timewarping approach. As can be seen, this approach performs quantitatively much better than the norm of distances approach. It might appear that thresholding is possible using this approach, but to increase the robustness with a high variability of gestures, we avoid this and pick the closest match instead. For each gesture, the matching and difference scores are also closer to each other for all users than those for norm of distances. Figure 9 (green) shows the matching accuracy when all the testing data was used for training, and Figure 11 (green) shows the matching accuracy with 4-fold cross-validation. As can be seen, this approach is more accurate than norm of distances.

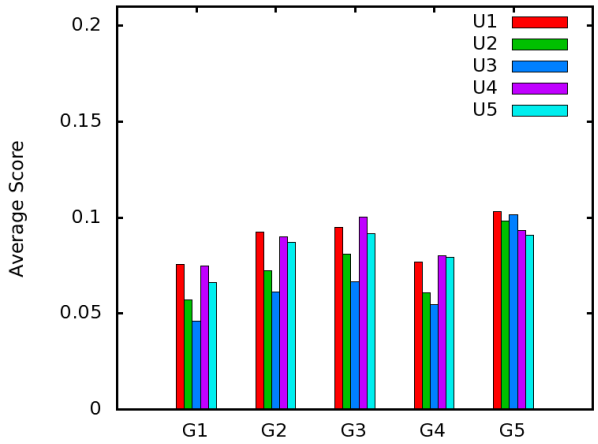

Figure 8. Matching scores for dynamic timewarping.

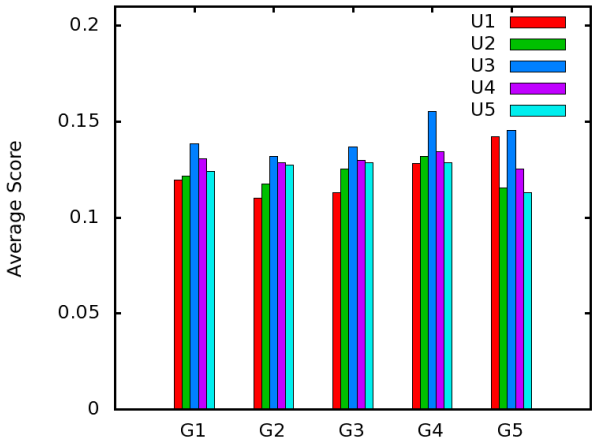

Figure 10. Difference scores for dynamic timewarping.

4.3. Multiclass SVM

We trained two multiclass SVM's, one on a small training data set and the other on a big training set. The small training set consisted of a subset of captured gestures whereas the big set contained not only the captured data but also numerous combinations of the input data where the last performed gesture was used for labeling. Figure 13 shows the matching scores for the bigger SVM and Figure 14 shows the difference scores for the bigger SVM. Note that the score values here correspond to probabilities of a given instance matching a particular candidate gesture. Also note in Figure 14 that the scale of the difference scores is different from that of the matching scores in Figure 13 because of the high accuracy of the bigger SVM. Figure 9 (blue and magenta) show the matching accuracies of the two SVM's when all the testing data was used for training, and Figure 11 (blue and magenta) show the matching accuracies of the two SVM's with 4-fold cross-validation. We notice that performance of the smaller SVM is comparable to that of the dynamic timewarping approach. Note that in Figure 11 the accuracy of the bigger SVM is smaller than that of dynamic timewarping for user 1 with 4-fold cross-validation because when a gesture is not in the training set the SVM fails to properly label it, while dynamic timewarping is more robust to unique gestures because of the pose alignment.

4.4. Real-time Interactivity

To demonstrate the viability of our method, we recorded a number of interactions with our real-time interactive physics simulation system. Figure 1 shows an example of a captured gesture along with a rendering of the real-time smoke simulation that results from matching this gesture to our training data. Figure 12 shows similar results for a number of frames of both smoke moving upwards and smoke

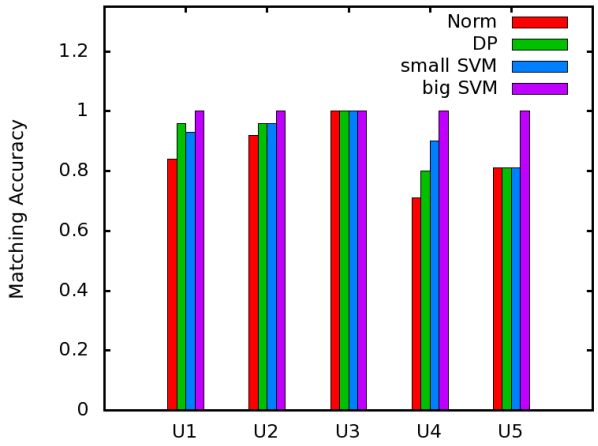

Figure 9. Matching accuracy for norm of distances, multiclass SVM's (trained with small and big training data sets) and dynamic timewarping when all testing data is used for training.

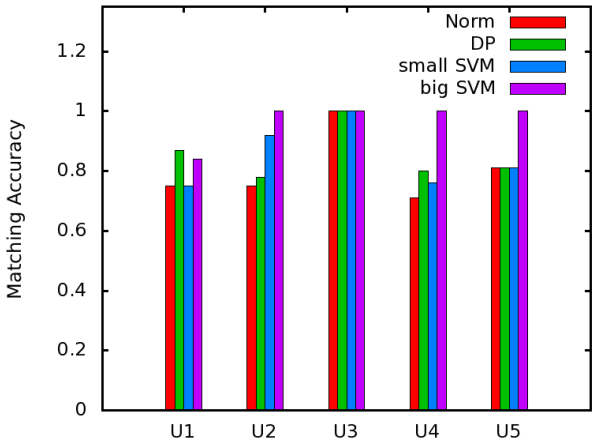

Figure 11. Matching accuracy for norm of distances, multiclass SVM's (trained with small and big training data sets) and dynamic timewarping with 4-fold cross-validation.

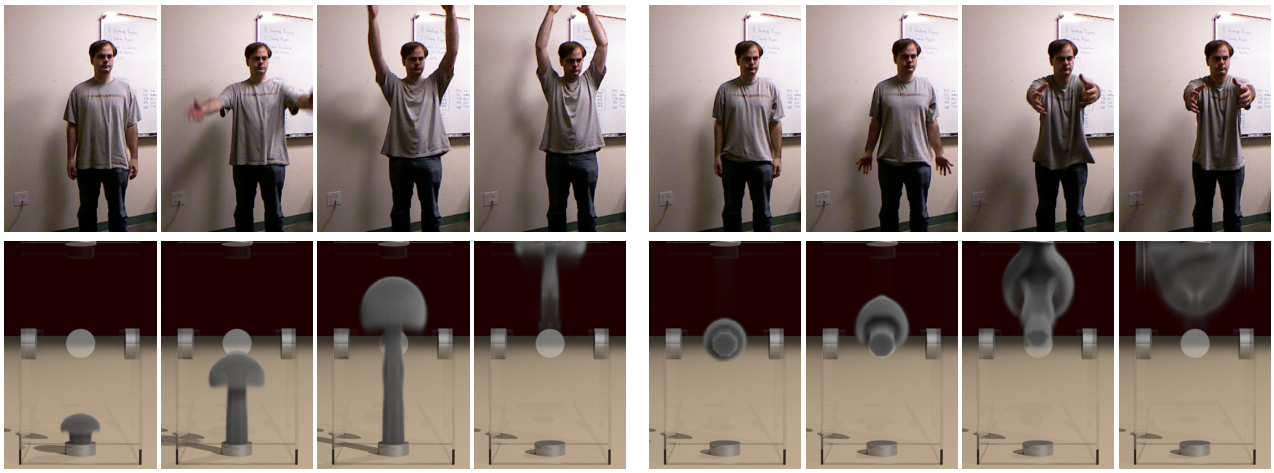

Figure 12. Two different gesture matches. (Left) shows the rising gesture which when matched will produce smoke rising as shown below. (Right) shows the forward gesture which when matched will push smoke forward through the domain as shown below.

moving forwards. Figure 15 demontrates the ability of our method to not only recognize gestures but use the smoke to track the hand motion. In this particular figure, the tracking algorithm was activated via a gesture. After the activation, the hand motion of the streaming skeletal motion data is used to move the smoke through the domain. In our simulations, the resolution is $40 \times 80 \times 40$, upsampled by a factor of 8 and with a framerate of 24 fps. Note the realtistic look that we are able to achieve using threading and upsampling despite the real-time nature of these simulations. Also note the accurate gesture matching and tracking achieved through our system (see video). In these figures, while the simulation is performed in real-time, the rendered smoke was added as a postprocess for visualization purposes and was not done in real-time. Also, in the video, the gestures are frozen after they are recognized by the system with the exception of the tracking gesture.

5. Conclusion and Future Work

We presented a number of algorithms for gesture recognition including norm of distances, dynamic timewarping and mutliclass SVM's. We demonstrated the effectiveness of these algorithms using a variety of evaluation statistics. We also presented a real-time simulation system which improves upon previous methods by adding parallelism as well as upsampling. Using these algorithms, we have been able to generate a real-time interactive system using Microsoft's Kinect. We envision that our system will have wide applicability, from special effects industries to education and software development. A number of directions are open for future work. Multiclass SVM's only work well when the training data accurately represents all possible gestures in the testing set. In contrast, dynamic timewarping is robust because of the pose alignment. An approach that combines the benefits of both these methods would need less training

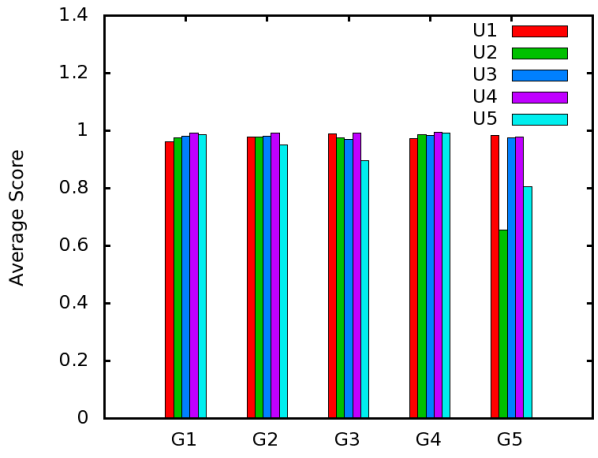

Figure 13. Matching scores for multiclass SVM.

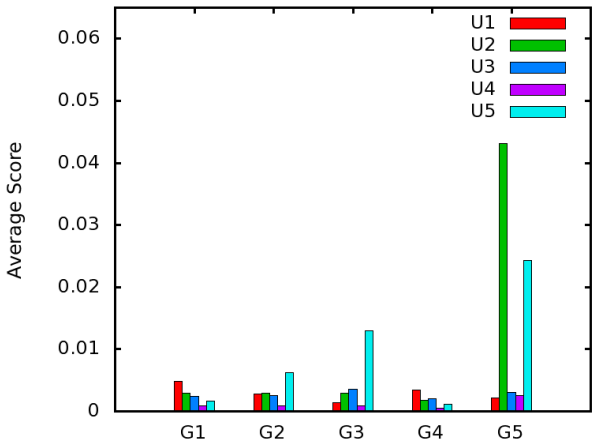

Figure 14. Difference scores for multiclass SVM. Note that the scale is different from that of the matching scores.

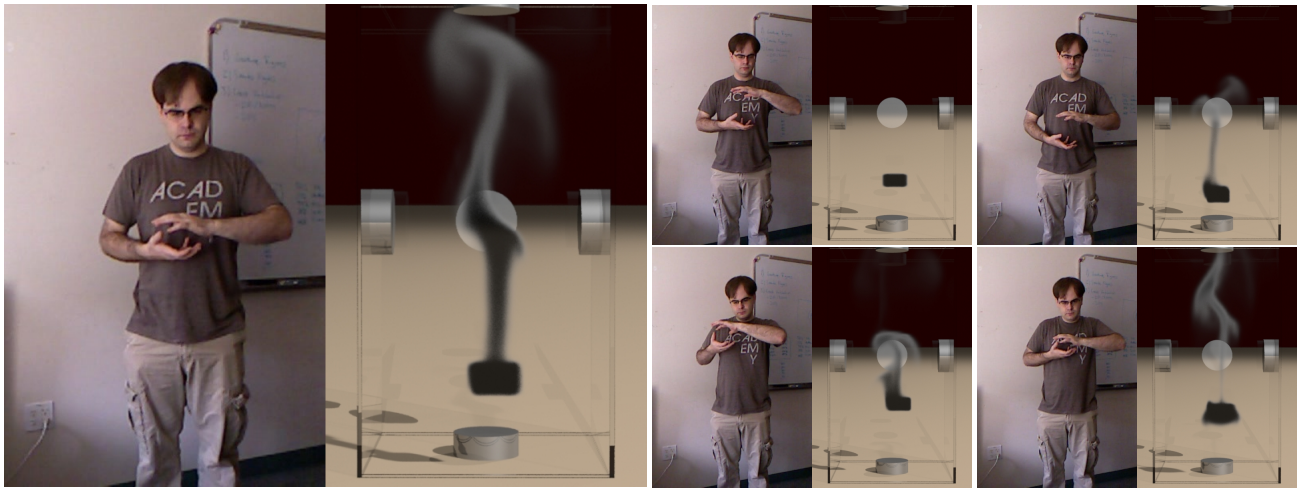

Figure 15. Tracking hand motion with kinect and the resulting simulation. Note that the hand motions are accurately tracked using our system.

data and would handle unique test cases, and is an avenue for future work. Moreover, gesture recognition can be made more robust to noise and outliers by using a motion curve approach [33, 4]. For computing similarity between two gestures, partial curve matching algorithms can be used [9]. Our current framework only supports the OpenGL visualization in real-time, and we plan to integrate our system with a real-time renderer such as the NVIDIA OptiX raytracer [3]. The performance improvements to our simulation systems can also be combined with higher resolutions simulations to create more realistic simulations with similar CPU run times. In the future, we would also like to integrate our system with other simulation solvers, such as water, cloth or fire.

References

[1] America's Army. `http://www.americasarmy.com/`.

[2] Microsoft Xbox Kinect. `http://www.xbox.com/kinect/`.

[3] NVIDIA PhysX. `http://www.geforce.com/Hardware/Technologies/physx`.

[4] J. Assa, Y. Caspi, and D. Cohen-Or. Action synopsis: pose selection and illustration. *ACM Trans. Graph.*, 24:667–676, July 2005.

[5] M. Berger and J. Oliger. Adaptive mesh refinement for hyperbolic partial differential equations. *J. Comput. Phys.*, 53:484–512, 1984.

[6] K. bo Duan and S. S. Keerthi. Which is the best multiclass svm method? an empirical study. In *Proceedings of the Sixth International Workshop on Multiple Classifier Systems*, pages 278–285, 2005.

[7] R. Bridson, J. Houriham, and M. Nordenstam. Curl-noise for procedural fluid flow. *ACM Trans. Graph.*, 26(3):46, 2007.

[8] A. Bruderlin and L. Williams. Motion signal processing. In *Proceedings of the 22nd annual conference on Computer graphics and interactive techniques*, SIGGRAPH '95, pages 97–104, 1995.

[9] K. Buchin, M. Buchin, and Y. Wang. Exact algorithms for partial curve matching via the fréchet distance. In *SODA 2009*, pages 645–654.

[10] M. Buhmann. *Radial Basis Functions: Theory and Implementations*. Cambridge University Press, 2009.

[11] C.-C. Chang and C.-J. Lin. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2:27:1–27:27, 2011. Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.

[12] F.-S. Chen, C.-M. Fu, and C.-L. Huang. Hand gesture recognition using a real-time tracking method and hidden markov models. *Image and Vision Computing*, 21(8):745 – 758, 2003.

[13] T. H. Cormen, C. Stein, R. L. Rivest, and C. E. Leiserson. *Introduction to Algorithms*. McGraw-Hill Higher Education, 2nd edition, 2001.

[14] M. Desbrun and M.-P. Cani. Smoothed particles: A new paradigm for animating highly deformable bodies. In R. Boulic and G. Hegron, editors, *Comput. Anim. and Sim. '96 (Proc. of EG Wrkshp. on Anim. and Sim.)*, pages 61–76. Springer-Verlag, Aug 1996.

[15] T. Dupont and Y. Liu. Back and forth error compensation and correction methods for removing errors induced by uneven gradients of the level set function. *J. Comput. Phys.*, 190/1:311–324, 2003.

[16] R. Fedkiw, J. Stam, and H. Jensen. Visual simulation of smoke. In *SIGGRAPH 2001*, 2001.

[17] N. Foster and D. Metaxas. Controlling fluid animation. In *Comput. Graph. Int.*, pages 178–188, 1997.

[18] Y. Gao, C.-F. Li, S.-M. Hu, and B. A. Barsky. Simulating gaseous fluids with low and high speeds. *Comput. Graph. Forum*, 28(7):1845–1852, 2009.

[19] A. Gruen and D. Akca. Least squares 3d surface and curve matching. *ISPRS Journal of Photogrammetry and Remote Sensing*, 59(3):151 – 174, 2005.

[20] C.-W. Hsu and C.-J. Lin. A comparison of methods for multiclass support vector machines, 2002.

[21] D. Kim, O.-y. Song, and H.-S. Ko. Stretching and wiggling liquids. In *SIGGRAPH Asia '09: ACM SIGGRAPH Asia 2009 papers*, pages 1–7, New York, NY, USA, 2009. ACM.

[22] T. Kim, N. Thürey, D. James, and M. Gross. Wavelet turbulence for fluid simulation. In *SIGGRAPH '08: ACM SIGGRAPH 2008 papers*, pages 1–6, 2008.

[23] L. Kovar and M. Gleicher. Flexible automatic motion blending with registration curves. In *ACM SIGGRAPH/Eurographics symposium on Computer animation*, SCA '03, pages 214–224, 2003.

[24] L. Kovar and M. Gleicher. Automated extraction and parameterization of motions in large data sets. In *SIGGRAPH 2004*, pages 559–568, 2004.

[25] L. Kovar, M. Gleicher, and F. Pighin. Motion graphs. *ACM Trans. Graph.*, 21:473–482, July 2002.

[26] M. Lentine, M. Aanjaneya, and R. Fedkiw. Mass and momentum conservation for fluid simulation. In *Proceedings of the 2011 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, SCA '11, 2011.

[27] M. Lentine, W. Zheng, and R. Fedkiw. A novel algorithm for incompressible flow using only a coarse grid projection. *ACM Trans. on Graphics*, July 2010.

[28] F. Losasso, F. Gibou, and R. Fedkiw. Simulating water and smoke with an octree data structure. *ACM Trans. Graph. (SIGGRAPH Proc.)*, 23:457–462, 2004.

[29] M. B. Nielsen, B. B. Christensen, N. B. Zafar, D. Roble, and K. Museth. Guiding of smoke animations through variational coupling of simulations at different resolutions. In *SCA '09: Proc. of the 2009 ACM SIGGRAPH/Eurographics Symp. on Comput. Anim.*, pages 217–226, 2009.

[30] T. Pfaff, N. Thuerey, A. Selle, and M. Gross. Synthetic turbulence using artificial boundary layers. In *SIGGRAPH Asia '09: ACM SIGGRAPH Asia 2009 papers*, pages 1–10, 2009.

[31] N. Rasmussen, D. Nguyen, W. Geiger, and R. Fedkiw. Smoke simulation for large scale phenomena. *ACM Trans. Graph. (SIGGRAPH Proc.)*, 22:703–707, 2003.

[32] W. Reeves. Particle systems - a technique for modeling a class of fuzzy objects. In *Comput. Graph. (Proc. of SIGGRAPH 83)*, volume 17, pages 359–376, 1983.

[33] A. Safonova, J. K. Hodgins, and N. S. Pollard. Synthesizing physically realistic human motion in low-dimensional, behavior-specific spaces. In *SIGGRAPH 2004*, pages 514–521, 2004.

[34] H. Schechter and R. Bridson. Evolving sub-grid turbulence for smoke animation. In *SCA '08: Proc. of the 2008 ACM SIGGRAPH/Eurographics Symp. on Comput. Anim.*, pages 1–7, 2008.

[35] A. Selle, R. Fedkiw, B. Kim, Y. Liu, and J. Rossignac. An Unconditionally Stable MacCormack Method. *J. of Sci. Comp.*, 35(2):350–371, 2008.

[36] A. Selle, N. Rasmussen, and R. Fedkiw. A vortex particle method for smoke, water and explosions. *ACM Trans. Graph. (SIGGRAPH Proc.)*, 24(3):910–914, 2005.

[37] J. Stam. Stable fluids. In *Proc. of SIGGRAPH 99*, pages 121–128, 1999.

[38] J. Stam and E. Fiume. Turbulent wind fields for gaseous phenomena. In *Proc. of SIGGRAPH 1993*, pages 369–376, 1993.

[39] S. B. Wang, A. Quattoni, L.-P. Morency, D. Demirdjian, and T. Darrell. Hidden conditional random fields for gesture recognition. In *Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on*, volume 2, pages 1521 – 1527, 2006.

[40] M. Wicke, M. Stanton, and A. Treuille. Modular bases for fluid dynamics. In *SIGGRAPH '09: ACM SIGGRAPH 2009 papers*, pages 1–8, New York, NY, USA, 2009. ACM.

[41] H. Wolfson. On curve matching. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 12(5):483 –489, may 1990.

[42] J.-C. Yoon, H. R. Kam, J.-M. Hong, S.-J. Kang, and C.-H. Kim. Procedural synthesis using vortex particle method for fluid simulation. *Comput. Graph. Forum*, 28(7):1853–1859, 2009.