CS231A Project Milestone: Real-Time Airbending

Mridul Aanjaneya
Stanford University
aanjneya@stanford.edu

Michael Lentine
Stanford University
mlentine@stanford.edu

Abstract

We propose to design a framework for real-time interactive physics-based simulation using state of the art motion sensing devices such as the Microsoft Kinect. The user would be able to interact with the system using simple gestures, where each gesture would have a predefined action associated with it. We would like our system to recognize logically similar gestures, i.e., gestures that represent variants of the same action. This will require novel pose descriptors and similarity metrics which would take into account the intrinsic relationships between different body parts rather than just the actual 3D embedding. Moreover, the real-time interaction would require fast algorithms for both physical simulation and gesture recognition.

1. Introduction

Real-time simulation has started gaining wide interest because of its potential applications to video games as well as special effects. Current state of the art algorithms for simulation attempt to approach real-time by coarsening the discretization of high-fidelity numerical methods. While it is necessary to achieve visual realism through simulations, it is also important to expand the domain of possible inputs for better interaction. With the recent development of the Microsoft Kinect, many researchers have started considering its possible applications to enhance the user experience.

2. Problem Statement

We wish to design a system for simulating natural phenomena in real-time which interacts with the user through gestures input using the Microsoft Kinect. We would like the system to recognize logically similar gestures, i.e., gestures that recognize variants of the same action [5]. For real-time gesture recognition, we would store all possibly similar gestures in an efficiently searchable data structure.

We plan on evaluating the performance of our gesture recognition algorithm on the training data we collected using our skeleton tracking algorithm (see Section 4.1), by dividing it into a training set and a test set. The evaluation metric would be the percentage of gestures for which the recognition succeeded. The performance of our interactive physics engine can be evaluated by computing the number of time steps taken per frame (which should ideally be 1) as well as the computation time per frame, which should be approximately 1/24 seconds for achieving real-time. Since we would be parallelizing our implementation, we would also like to compare the total speedup achieved against the number of threads used to decompose the domain.

2.1. Gesture Recognition

Gestures are perhaps the most intuitive way of interacting with a given system. While gesture recognition has been an active area of research, most available systems are constrained to recognizing numerically similar gestures, i.e., the corresponding skeletal poses are roughly the same. This contradicts the very definition of a gesture. For example, if one wants to shoot a fireball to the right and to the left, the performed action remains the same, but the user shoots in two different directions. Current algorithms will incorrectly interpret these actions as two different gestures. Hence, one needs to design descriptors that can encode the semantic meaning of an action and intrinsic relations between various body parts, as opposed to just the extrinsic 3D embedding. Moreover, one needs to learn a metric on the space of such descriptors for accurate gesture matching and for real-time recognition, one needs to design a compact data structure for storing similar gestures which is efficiently searchable [5]. If time permits, we also wish to explore the problem of sampling gestures for encoding them compactly while still preserving their semantic meaning [1].

2.2. Real-time Interactivity

While gesture recognition is necessary for an interactive simulation, real-time algorithms for physically based simulations also need to be developed. In order to achieve plausible results, many current methods rely on fine discretizations in both space and time, making these techniques impractical for real-time applications [3]. To alleviate these problems, we plan on implementing methods that reduce the cost of individual steps in a traditional simulation algo-

rithm as well as techniques for conserving physical quantities and thus allowing for accurate coarser discretizations.

3. Technical Approach

Our method consists of two main parts, real-time gesture recognition and real-time interactive simulation.

3.1. Gesture Recognition

There are two main approaches to gesture recognition. 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 [6, 4, 5]. 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 [9, 1]. Individual gestures are then matched using state of the art curve matching algorithms [2]. We adopt the former approach and use the framework proposed by Kovar and Gleicher [5]. If time permits, we wish to investigate the latter approach as well.

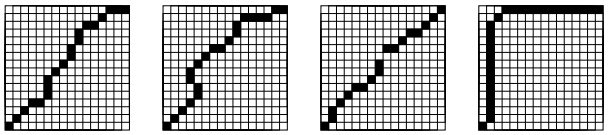
Each pose is represented as a point cloud derived from the skeleton created from cylinders (see Section 4.1). Distance between two poses is the weighted sum of squared distances between corresponding points $\mathbf{p}_i$ and $\mathbf{p}'_i$ in the two point clouds:

$$d(\mathcal{P}_1, \mathcal{P}_2) = \min_{\theta, x_0, z_0} \sum_i ||\mathbf{p}_i - \mathbf{T}_{\theta, x_0, z_0} \mathbf{p}'_i||^2 \qquad (1)$$

The transformation $\mathbf{T}_{\theta, x_0, z_0}$ rotates a point $\mathbf{p}$ about the Y (vertical) axis by θ degrees and then translates it by (x_0, z_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, as described in [6]. For computing logically similar gestures, we start by computing numerically similar gestures, and then use them as new queries for finding more distant gestures. Two gestures are numerically similar if the corresponding frames have similar poses and related gesture events are easily recognizable. The set of pose correspondences should form a continuous, monotonically increasing, non-degenerate mapping between frames, which is termed as *time alignment* [5] (see Figure 1). An optimal time alignment that minimizes the total distance between matched poses can be computed using dynamic programming [4]. For real-time recognition, we store similar gestures in a *match web* [5] which is an efficient search structure.

3.2. Real-time Interactivity

Our method starts by using [3] for smoke simulation. We then add additional performance using [8] which downsamples the resolution for the expensive parts of a simulation



(a) (b) (c) (d)

Figure 1. Time alignments must be continuous, monotonic and non-degenerate: (a) legal, (b) nonmonotonic, (c) discontinuous and (d) degenerate.

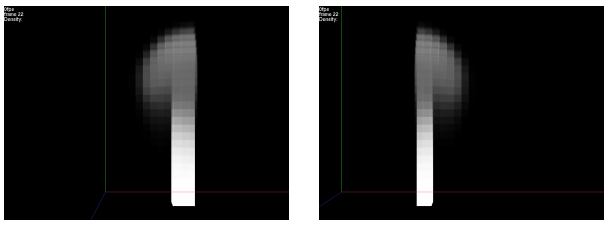


Figure 3. (Left) output of only the thread associated with the left side of the domain and (Right) a thread on the right side of the domain.

while keeping a large resolution to preserve details. We further improve upon this by adding [7] which allows for a single time step to be taken for every frame. Because of these techniques we can now run grid based simulations with real-time performance. Our goal is to further improve this performance and achieve significantly higher resolution simulations while maintaining the real-time constraint.

We approach this problem by using two methods. First step is to add parallelism to these simulations. We achieve this 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 3). 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. To achieve this we first break up our problem by dimension and apply the filter g on each dimension.

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

4. Preliminary Results

Using Microsoft's Kinect is central to our problem. Therefore we start by connecting the Kinect to a PC and use the openni device drivers to read data. Input data has color and depth information. We can then use a skeleton tracker such as that provided by NITE. This provides us with a set of joint angles as shown in Figure 4 (Left).

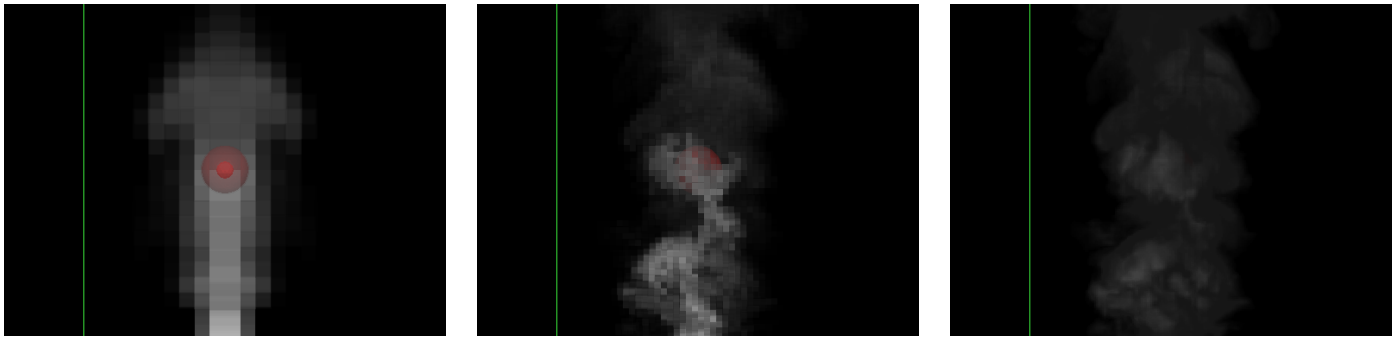


Figure 2. (Left) using prior methods from [8, 7] in real time. (Center) our new method incorporates parallelism allowing for higher resolution simulations. (Right) our new method after convolving with an upsampling filter. Note the improved visual fidelity.

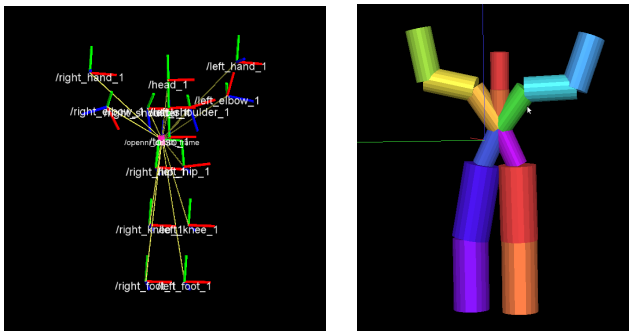


Figure 4. (Left) capture of joint angles using Kinect data and (Right) a skeleton created from cylinders.

4.1. Gesture Recognition

Once the joint angles were available, we converted them into an appropriate skeleton which would be better suited as input data for our gesture recognition algorithm. To compute the skeleton we constructed a set of cylinders for each bone. This was achieved by calculating the differences between positions of each joint and then using the relative joint rotations to compute the orientation of the bones. We use OpenGL to display the results of our skeleton tracking algorithm (see Figure 4 (Right)). We plan to collect training data for gesture recognition by using our skeleton tracking algorithm and manually cropping frames for each gesture. This data would then be used as input for our algorithm.

4.2. Real-time Interactivity

Real-time interactive simulations form the other part of our problem. As described in Section 3.2, we have implemented prior algorithms and added parallelism which drastically improves their performance. Using 12 threads we can improve the performance of our algorithm from a grid of resolution $20 \times 40 \times 20$ to a grid of resolution $50 \times 100 \times 50$, as shown in Figure 2 (Left) and (Center). To further increase the visual fidelity of the simulation we applied a convolution filter that upsamples the resolution of the simulation using first order interpolation. The results of this filter, as compared to the results of just our real-time simulation, are shown in Figure 2 (Center) and (Right).

References

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

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

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

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

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

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

[7] 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.

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

[9] 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.