Simulation of Card Spring Process

Ruoning Ren Huangshuai Shuai Jiahui Xi

Abstract

The Card Spring is probably the most commonly used card flourish and cardistry technique in existence. This beautiful visual of the playing cards jumping from hand to hand is one of the card magic basics that every beginner card magician should learn. As an engineer, we are looking forward to developing an explicable DEB(Discrete Elastic Beam) model to simplify and simulate the whole process by Matlab, also future possible rendering by blender

1 Background

When we watch some gambling-related movies, we always see some fancy moves that can be designed to show the gambler's superb gambling skills. For example, multiple shuffle methods, one-handed or two-handed cut. Some of the cardistry skills are easy to learn, since they may not need much practice. However, there is one trick that looks pretty cool, much cooler than other skills, and really hard to learn. It is called card spring.

Studying the model of the whole process can help us better understand this cardistry and more thoroughly practice it. After getting familiar with knowledge about simulation of discrete elastic structures, we believe that even if we are not able to show this trick by our hands, we can still simulate it with the help of matlab and rendering software. Such a simulation may be helpful in some anime creation situations, and the physical process can be deeply understood by this simulation..

Moreover, we might take the advantage of implicit integration and direct constraint satisfaction introduced by David and Andrew [1]. In this method, mass modification will be applied to constrain node acceleration and direction.

2 Basic Principle

2.1 Assumption

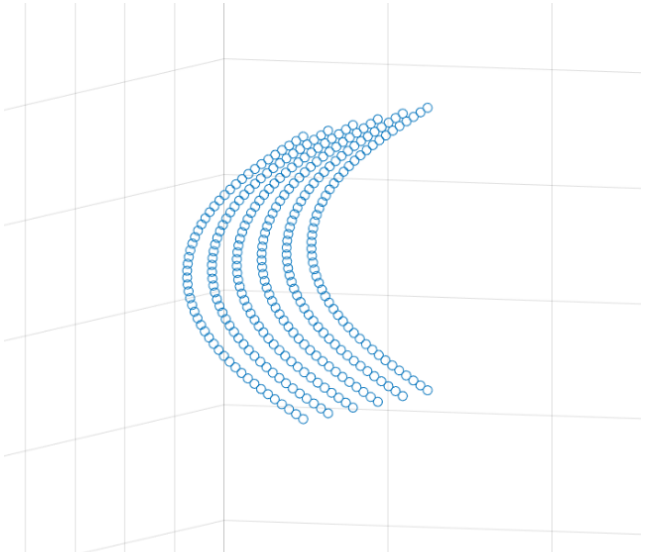
Since the plate model of the card is relatively complex, we make some assumptions about the card.

Firstly, we suppose that the mass distribution of the card is uniform. Then, the distribution of bending force on the top and bottom edge of the card is also uniform, and the force components are only parallel to the x and y axis. Based on these assumptions, we can simplify the card elastic model to a stack of Discrete Elastic Beam (DEB) models.

2.2 Three Phase Stage

a. Phase I: bending

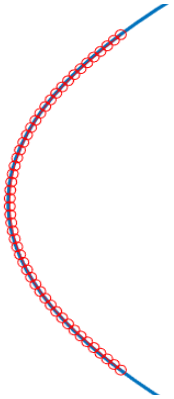
Firstly, we will assume the card as a parallel beam with a fixed top node in x,y direction and fixed bottom in x-direction. To mimic the effect of actual card bending, we first give a perturbation in the x-direction to a node in the middle of the card, and then apply an upward force parallel to the y-axis to the node at the bottom of the card. When this force is constant, the card model will have oscillation, so use a PD controller for this force to make the card reach the equilibrium state as soon as possible.



b. PhaseII : contacting

At this stage, the nodes that hold the card will be released, and all external forces applied to the card will disappear, so that the card will stretch due to the release of potential energy. During the stretch, the card hits the stack and thumb, gaining reaction force from them and being ejected.

The constraint by stack will be fitted by a quadratic function.



c. Phase III: after contact

In Phase III, the card will no longer receive any contact force and artificial external force, but will fly in the air under the action of damping force and gravity.

To summarize, we simulate the whole card spring as three phase stages: Phase I,II,III for only the card on the top of the deck, which briefly simplifies the entire process into a discrete elastic beam model. For phaseI, bending the card until it reaches a designated equilibrium form. Also, the contact force and constraint model is also being considered in phase II to maximumly reproduce the actual physical phenomenon. Then, in phase III, just release the card in the air, the card will Free oscillation and motion after leaving from boundary contact with the effect of gravity and air friction.

3 Methodology

In the current stage, we are using a parallel 2-D beam structure to emulate a 3-D card structure. Discrete elastic rods [2] methods are used for our simulation.

3.1 Discrete Beam Formulation

As we learned from class, based on Newton's second law, $F_i = m_i \ddot{q}_i$,we develop the following equation:

$$f_i = m_i \frac{q_{i(t_{k+1})} - q_{i(t_k)}}{dt^2} - C_i \frac{q_{i(t_{k+1})} - q_{i(t_k)}}{dt} + \frac{\partial}{\partial_{q_i}} (E_i^b + E_i^s) - F_i^e$$

The second section on the right side of the equation, damping force, is only counted in Phase III, and the external force is only exerted to the card in Phase I. Considering that constraint is defined in Phase2, f_i can be seen as the reaction force, which must equal to zero when in Phase I and Phase III.
For simulation using the DER method, The right side of the equation should add a twist force section.

3.2 Newton-Raphson iteration

We use Newton-Raphson iteration to solve each time step:
$q := q - J/f$

where, J, the Jacobian, is calculated by $J_{ij} = \frac{\partial f_i}{\partial q_j}$.
The Jacobian is composed by the Hessian matrices of elastic energies in each section of the structure, the acceleration sections, and the partial derivative of external force. However, external force in Phase I has no relation to position of the node, So it won't be calculated in our simulation, i,e, $\frac{\partial Fe}{\partial q_j} = 0$.

3.3 Mass Modification

Mass Modification is implemented for defining the constraint in simulation modeling. This method requires the inverse of the mass matrix, $\mathbf{M^{-1}}$ instead of $\mathbf{M}$.

For the case of a single node, $\ddot{q}_i = (1/m_i) f_i$ describes the node's acceleration. When the node is needed to be restricted of motion, that is to keep its velocity from changing, $1/m_i$ is taken to be zero. The corresponding term inside the inverse mass matrix is made zero accordingly.

When imposing a constraint in the *i-th* node, the mass matrix is modified in the *i-th* term on the diagonal. Each term on the diagonal has a size of 2x2 corresponding to the *x-component* and the *y-component* (as modeled in the DEB approach). To add constraint along a certain axis, the corresponding component needs to be set to zero. Then, only the accelerations on the unconstrained nodes or unconstrained axes of a node is considered.

Suppose a constraint has components on both x and y axes, the node is restricted of accelerating along $\boldsymbol{P}$ (a unit vector with x and y components), the modified inverse mass matrix is defined as $1/m_i(I - P_i P_i^T)$.

When obtaining the Δv_i consistent with the constraint, a velocity component z_i along the constraint $\boldsymbol{P}$ needs to be imposed to balance the equation of motion constituted by Newton's 2nd law.

4 Further Improvements

Since the constraint plane is fitted by a quadratic function, the constraint direction can be changed by the desired contact position of the card, and the desired contact position is hard to find. As a result, we should actually optimize the contact model of a card in Phase II.

In addition, video or photo generated in matlab is not quite enough for a reliable animation. So other rendering methods will be applied to this simulation .

5 Resource

We decided to use Matlab to realize those simulations. And the following library may help us :

Three.js - we will use consoles created by others to display our results, based on three.js.

Other libraries may also be found required in the future.

Link to the Presentation Video:
https://www.youtube.com/watch?v=FtgqvkCBMIU

6 Reference

[1] Baraff, David, and Andrew Witkin. "Large steps in cloth simulation." In *Proceedings of the 25th annual conference on Computer graphics and interactive techniques*, pp. 43-54. 1998.

[2] Bergou, M., Wardetzky, M., Robinson, S., Audoly, B., and Grinspun, E. Discrete elastic rods. In *ACM SIG GRAPH 2008 Papers* (2008), SIGGRAPH '08, pp. 63:1–63:12.