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 [1] 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[2] 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 Phases of Motion

a. Phase I: Bending

Firstly, we assume the card as a single 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 beam model will oscillate and remain in motion for a period of time, to imitate the fast gripping to hold the card in place, a PD controller is implemented in order to make the card reach equilibrium state quickly..

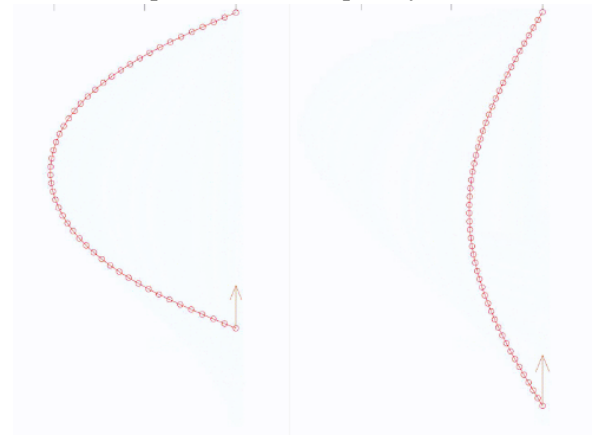

Figure 1. Beam compression
(left with PD control, right without PD control)

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. The thumb will be defined as an extended boundary segment at the bottom of the

beam that will remain in contact with the last node until release.

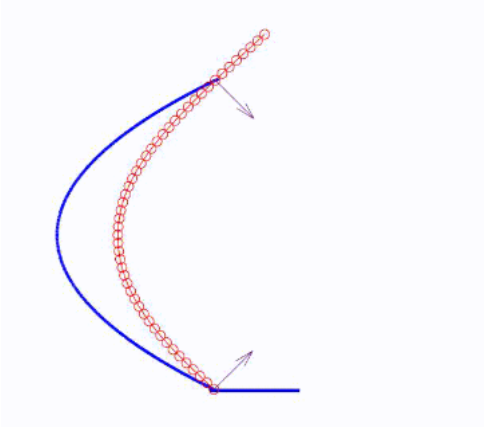

Figure 2. Contact Boundary with normal force vector

c. Phase III: Post-contact

In Phase III, the card will no longer receive any contact force and artificial external force, but will oscillate freely 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 phase I, 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 [3] methods are used for our simulation.

3.1 Discrete Beam Formulation

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 accounted for in Phase III, and the external force is only exerted to the card in Phase I.

Considering that constraint is defined in PhaseII, 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:

$$f_i = \frac{m}{\Delta t}\left(\frac{q_i^{k+1} - q_i^k}{\Delta t} - q_{d_i}^k\right) + \frac{\partial E_{\text{potential}}}{\partial q_i} + c_i \frac{q_i^{k+1} - q_i^k}{\Delta t} + F_{fi} + F_{ei}$$

$$\Delta q = J \backslash \mathrm{f}, \; q_i^{k+1} := q_i^{k+1} - \Delta q$$

where, J, the Jacobian, is calculated by

$$J_{ij} = \frac{\partial f_i}{\partial q_j} = \delta_{ij} \frac{m}{\Delta t^2} + \frac{\partial^2 E_{\text{potential}}}{\partial q_i \partial q_j} + C_i \frac{\delta_{ij}}{\Delta t}$$

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[2] 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 z 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.

The jacobian based on the Mass Modification method is calculated with

$$J = \frac{\partial f}{\partial q} = I \cdot \Delta t + I \cdot \Delta t \cdot \mathrm{W} \cdot J_{\text{initial}} + I \cdot \Delta t \cdot \mathrm{CW}$$

To update the position of the nodes, the same logic as the Newton-Raphson iterative method is employed

$$\Delta q = J \backslash \mathrm{f}, \quad q_i^{k+1} := q_i^{k+1} - \Delta q$$

3.4 Constraint Definition

A constraint boundary similar to the grasping palm of the bending motion applied on the stack of cards is relatively complex. Here, a simplified scenario is applied similar to the 'pure sliding' contact modeling by Huang et al[4]. To determine the constraint boundary, a 2nd degree polynomial is used to apply the bent card behind the springing card as shown in Figure 3 as the blue curved line. During the release motion, or the motion considered in Phase II, where the card rebounds from the constraint boundary behind it, the mass modification method is implemented to set the constraint conditions of nodes in contact with the constraint boundary. Without constraint, these nodes will pass through the constraint boundary. To tackle this problem, q_{old} (or the position of the previous iteration) of the node supposed to be in contact is used in prediction of the node's position, $q_{new}^{predict}$, in the current iteration when the constraint is ignored. The satisfactory contact position of this node is calculated with two crucial conditions - the contact point,$q_{new}^{correct}$, lies on the tangent line to the quadratic curve; the line passing through $q_{new}^{predict}$ and $q_{new}^{correct}$ is perpendicular to the tangent through the contact point.

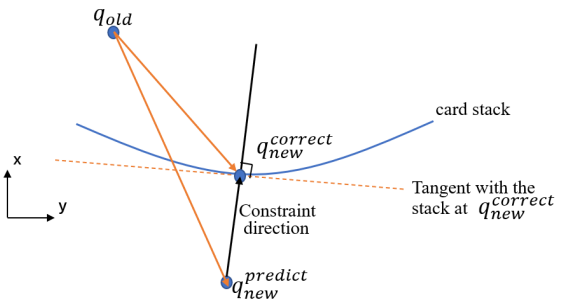

Figure 3. Constraint Definition

The above procedure is key to applying the mass modification method. The normal vector P at the i-th node in the modified mass matrix below.

$$W = M^{-1} \cdot (I - P_i P_i^T)$$

where, $M^{-1} = 1/m_i$

Pseudo code for functions *Compute Stack* and *Judge and Change* that verifies whether node passes constraint boundary:

Algorithm 1 Compute Stack & Judge and Change

Require: q_{new} ▷ Degree of Freedom
Require: N ▷ Number of Nodes
Ensure: $q_{con}, thumb$ ▷ updated constraint
Ensure: $stack, stack\_bound$ ▷ Quadratic Coefficients
  function COMPUTE_STACK
    for $i \leftarrow 1 : \mathrm{N}$ **do**
      $q_{con} \leftarrow q_{new}(2*i-1:2*i)' + [-0.0003, 0]$
    end for
    $stack \leftarrow polyfit(q_{con}(:,2), q_{con}(:,1), n)$
    $thumb \leftarrow q_{con}(1,2)$;
    $stack_{bound} \leftarrow |q_{con}(1,2), q_{con}(N,2)|$;
  end function

Require: $q_{new}\ q_{old}$ ▷ Degree of Freedom
Require: F ▷ Force
Require: $ground$ ▷ Ground costraint
Ensure: $commap$ ▷ Constriant of each node
Ensure: $recal$ ▷ Correcotr
  function JUDGE AND CHANGE(q_new, q_old, F)
    $recal \leftarrow 0$
    $variance \leftarrow 0.00001$
    for $i - 1 : N$ **do**
      $node\ ith\ y\ position \leftarrow q\_new(2*i)$
      if Node i constraint $==$ 0 **then** ▷ Add Constraint
        if $node\ ith\ y\ position < ground$ **then**
          $commap(i) \leftarrow 1$
          $recal \leftarrow 1$ ▷ unit normal vector at contact point
        end if
      end if
    end for
    for $i - 1 : N$ **do**
      $node\ ith\ y\ position \leftarrow q\_new(2*i)$
      if Node i constraint $==$ 1 **then** ▷ Remove Constraint
        $frec - F(2*i-1:2*i)$
        $fnormal - dot(frec, [0,1])$
        if $node\ ith\ y\ position < ground - 2e^{-}6 \&\& fnormal <= 0$ **then**
          $commap(i) \leftarrow 0$
          $recal \leftarrow 1$ ▷ unit normal vector at contact point
        end if
      end if
    end for

    return $commap, recal$
  end function

3.5 Material Properties

The material properties assumed for the simulation are listed as follows: static friction is introduced in Phase III only. Phase I does not consider friction. Phase II assumes slipping, therefore, static friction is assumed to be zero. The friction is calculated with the following formula

3.6 Friction Setting

Static friction is introduced in Phase III only. Phase I does not consider friction. Phase II assumes slipping, therefore, static friction is assumed to be zero. The friction is calculated with the following formula:

$$f = dot(\mu f_{reac}, n_1) \cdot n_2$$

$$\text{where, } n_1 \cdot n_2 = 0$$

$$\text{and } (n_2 \cdot qd_{old}) > 0$$

n_1 is a unit vector along the constraint direction and μ is the coefficient of friction of the ground contact.

For the final simulation with a stack of four cards, the cards eventually get in contact with the ground surface and reach the end of motion due to static friction. (note. the coefficient of friction is selected based on simulated ground surface material, in the case of the simulation, μ is set to 0.5 for a wooden tile ground surface)

3.7 Animation Rendering

Instead of animating a single beam, to better imitate the card in all phases, the positions of the nodes in (x,y) coordinates are transformed into (x,z) coordinates. Each beam is duplicated 3 times and evenly spaced out into the positive y-direction.

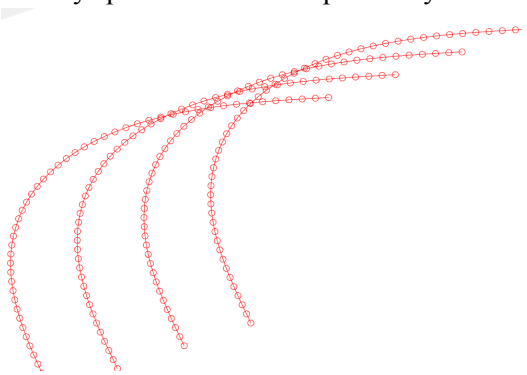

Figure 4. Four Beam Representation

To construct a representation of the card, the edge lines of the card are displayed in red, forming a rectangular frame, such that the deformation and recovery of the card during the flex-and-release motion is highlighted.

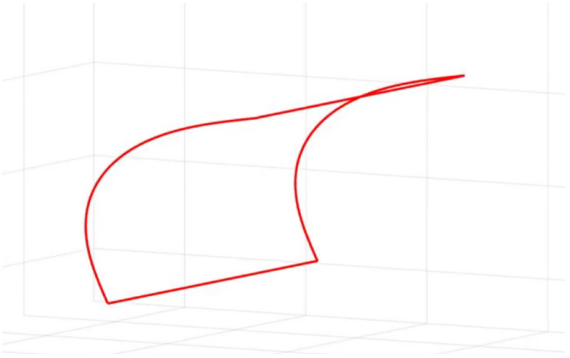

Figure 5. Edgeline Representation

For the final render of the animation, a simulation of a stack of four cards springing out in a sequential fashion is presented. The animation demonstrates four cards, the Ace of Spades, two of Diamonds, three of Clubs and four of Hearts, go through the three phases and eventually end their motion due to static friction from the ground surface.

Figure 6. Final Render

The visual effects of card faces, ground surface material and lighting is added with the surface plot function in MATLAB[5].

4 Future Work

In the future, further improvements will be implemented. At the current stage, the contact and collision between cards have yet to be accounted for. Later, the collision between cards will be added in Phase III just before the cards reach the ground surface for a more realistic simulation. Apart from adding better interaction dynamics to the simulation, a more sophisticated modeling technique of Discrete Elastic Plates will be introduced so that the simulation will be applicable to other hand configurations where twisting is imposed on the cards in the simulated phases. The current animation rendering uses the 'surface' function in MATLab, we plan to create better visual rendering with Three.js in the near future.

5 Reference

[1] Jawed, M. K., Novelia, A., and O'Reilly, O. A primer on the kinematics of discrete elastic rods. Springer-Verlag, 2018

[2] 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.

[3] 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

[4] Huang, W., Huang, X., Majidi, C. et al. Dynamic simulation of articulated soft
robots. Nat Commun 11, 2233 (2020). https://doi.org/10.1038/s41467-020-15651-9

[5] Surface plot - MATLAB. (n.d.). from https://www.mathworks.com/help/matlab/ref/surf.html