Final research project: Cloth Simulation(Computer Graphics, COMP8610-2024)

Pan Zeng	Zihan Yuan	Jiawei Ye
u7799442	u7773880	u7574419

The Australian National University

Abstract

This project is based on the mass-spring model to simulate deformable cloth and compares the effectiveness of various numerical formulas while implementing collision detection functionality. Initially, the cloth object is approximated as a deformable surface comprised of a network of masses and springs, with its movement evaluated through the numerical integration of basic dynamics laws. When high stresses are concentrated in a small surface area, the local deformation becomes unrealistic compared to fabric deformations. We use constraints to adjust the model accordingly. Secondly, when the cloth collides with rigid objects, we adjust the velocities of the masses to handle the collision. Finally, we simulate the process of cloth deforming due to natural wind blowing against it.

Keywords: Physically-based models, deformable surfaces, cloth animation, Runge Kutta, Verlet integration, collision solution•

1 Introduction

In this study, we expanded upon the existing mass-spring model to enhance our simulation methods. Beyond simply reimplementing the model with constraint limitations and employing the Euler method for numerical simulations, we integrated advanced Runge-Kutta and Verlet integration techniques. These methods are well-suited for systems experiencing rapid dynamic changes and complex interactions, offering enhanced precision and stability in simulation outcomes.

Moreover, we enhanced the model's functionality by implementing collision detection using Axis-Aligned Bounding Box (AABB) technology. This approach significantly streamlines the collision detection process for numerous particles, reducing computational load and enhancing the rendering speed.

This project not only implements functions such as cloth simulation and constraints described in the original paper; it also introduces functions such as numerical methods and collision detection that minimize errors and enhance stability, significantly broadening the applicability and efficiency of the model in dynamic simulations.

2 Motivation and Problem Statement

The realistic simulation of cloth in computer graphics presents a significant challenge due to the complex dynamics of fabric behaviour and interaction with environmental forces. Traditional mass-spring models often fail to perfectly mimic the non-elastic properties of textiles, especially under high

stress or when interacting with rigid objects, leading to unrealistic deformations. These limitations hinder the models' applicability in practical scenarios such as virtual fittings where visual realism is critical.

Our project aims to overcome these limitations by enhancing traditional simulation methods with advanced numerical techniques and improved collision detection mechanisms. By integrating Runge-Kutta and Verlet integration methods, we hope to achieve more accurate and stable simulations, particularly when the cloth interacts with dynamic environmental elements like wind or physical objects.

3 Previous Works

In "Deformation Constraints in a Mass-Spring Model to Describe Rigid Cloth Behavior", Xavier Provot (1995)(1) explored the limitations of traditional elastic models in realistically simulating the behaviour of woven fabrics in computer graphics. He introduced an innovative modification to the mass-spring model by integrating non-elastic properties into the model, which helps prevent unrealistic deformations under high-stress concentrations. He critiqued the 'super-elastic' tendencies of standard elastic models, which resemble rubber more than the actual fabric, arguing that they do not capture the true stiffness and constraints of textiles. To address this, a dynamic inverse procedure is integrated into the model, enhancing the model's ability to closely mimic the physical properties of textiles and thereby increasing the realism of cloth animation. Furthermore, his method tackled the computational inefficiencies that exist in traditional models, offering a solution of lower computational costs but higher quality simulations.

4 Our Method

4.1 Reproduce the paper

4.2 Cloth structure

First, we set up a mass-spring model to simulate a cloth. The model contains three different springs: structural, shear and flexion springs, as described by Provot(1995)(1). The model is represented as a grid of virtual masses(nodes) connected by springs.

4.2.1 Springs1

1. Structural Springs: Springs connect its neighborhood nodes to maintain the structure of the cloth under different forces.
2. Shear Springs: These springs link diagonal masses to prevent shear deformations, ensuring the cloth won't experience unrealistic distortions.
3. Flexion Springs: By connecting masses two nodes apart, this springs guarantee the cloth to fold and bend realistically.

4.2.2 Dynamics and Forces

The force working on the masses are based on fundamental law of dynamics, including internal and external forces.

1. Spring Force: The force is given by Hooke's law:

$$F_{spring} = -k \cdot l \tag{1}$$

   where k is a constant representing the stiffness of the spring, and l is the current length of the spring.

2. Damping Force: To simulate energy loss in real world, the damping force is given by:

$$F_{damping} = -C_{damp\_coef} \cdot v \tag{2}$$

   where C_{damp_coef} is the damping coefficient and v is the velocity of the mass.

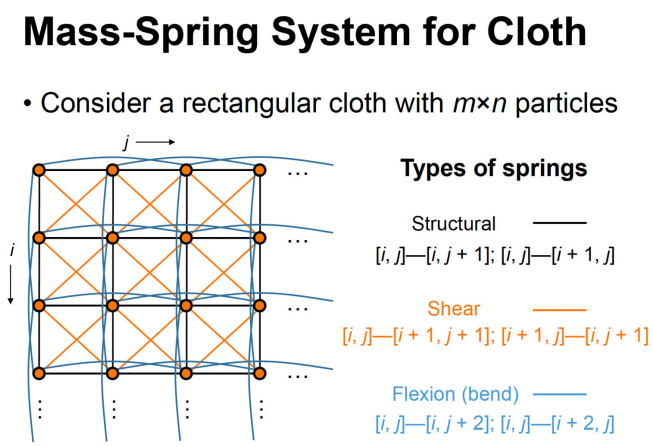


Figure 1: Mass-spring system from ANU COMP8610 lecture slides

3. Gravity: Since every mass has weight m, we add gravity by using:

$$F_{gravity} = m \cdot g \tag{3}$$

where g is the acceleration of gravity.

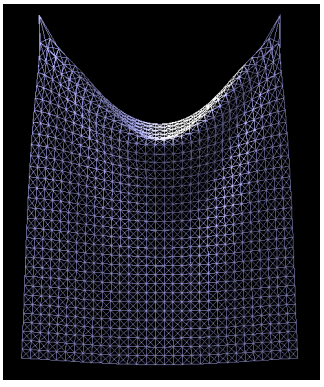
4. Aerodynamic Force: The influence of air resistance is simulated by the Aerodynamic Force:

$$F_{fluid} = C_{fluid} \cdot (u_{fluid} - v) \cdot n \tag{4}$$

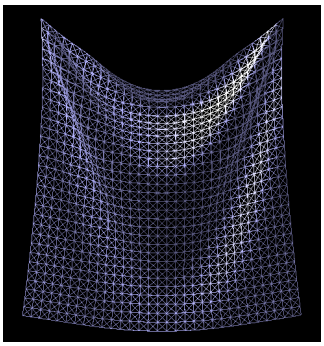
where C_{fluid} is the fluid coefficient, u_{fluid} is the velocity of air(equals to 0 when there is no wind), v is the velocity of the mass, and n is unit normal on the surface.

4.3 Deformation rate

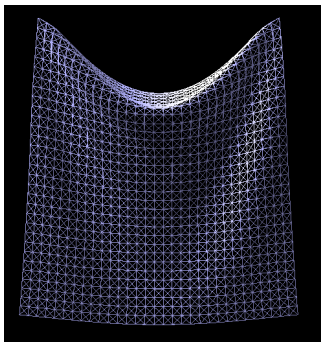
To make the cloth look more like real firmer fabric such as woven fabrics and avoid "super-elastic" effect mentioned in paper(1), we implemented deformation constrains by setting deformation rate to 10%. This means that, if the length of a spring is longer than its original length by more than 10%, then the spring's two masses will be applied a dynamic inverse process until its deformation rate precisely equals 10%. Figure2 shows the result we get. In Figure 2a, we can clearly see the springs of upper corners of cloth are excessively stretched. And the "super-elastic" effect is successfully suppressed after we apply constraints on different type of springs in Figure 2b and 2c.



(a) Without Constraint



(b) Constraint applied to 'Structural' Spring



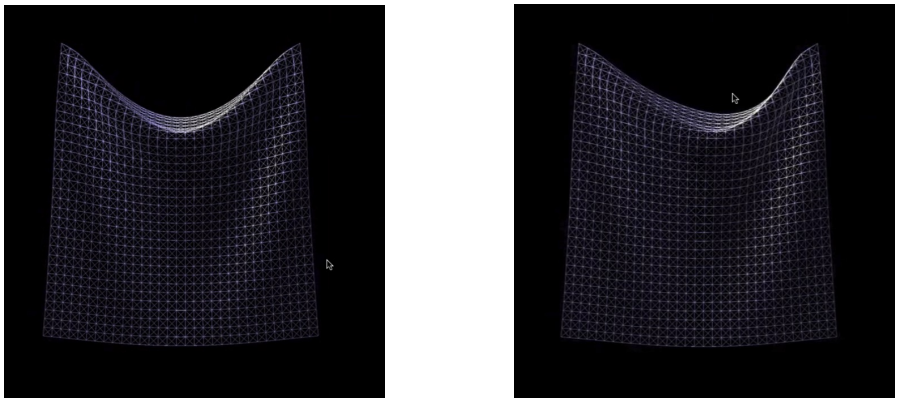
(c) Constraint applied to 'Structural' and 'Shear' Spring

Figure 2: Deformation of cloth

4.4 Dynamic Wind Interaction

In our cloth simulation, the wind interaction is triggered by cursor clicks and movements, providing a realistic and interactive visualisation of how wind affects fabric. In the `cursor_pos_callback` method, the `'windBlowing'` flag is checked to ensure the simulation responds only when appropriate. The wind's direction and force are calculated from the cursor's displacement relative to a predefined starting point, altering the wind direction based on this movement.

Each cloth mass coordinate point is transformed into screen space to evaluate its distance from the cursor. If within the defined radius, a decay feature based on the cosine of the distance applies a diminishing wind force to the mass. This setup creates a dynamic simulation where the cloth's reaction varies with both the strength of the wind and the proximity (i.e., the radius) of the wind, enhancing the visual and interactive realism of the model.



(a) Original Mesh (b) Mesh with wind

Figure 3: Dynamic Wind Interaction

4.4.1 Semi-Implicit Euler Method

Implicit Euler uses the velocities of the next timestep to estimate the positions of the particles.

$$a = F/m \tag{5}$$

where a is the acceleration of mass.

$$v_{t+1} = v_t + a_t \cdot \Delta t \tag{6}$$

where v_{t+1} is the speed of mass at next timestep.

$$x_{t+1} = x_t + v_{t+1} \cdot \Delta t \tag{7}$$

where x_{t+1} is the position of mass at next timestep.

4.5 Improvement

4.5.1 Runge-Kutta

The Runge-Kutta method is a higher-order numerical integration technique with lower computational errors than implicit Euler. In this project, we employed the fourth-order Runge-Kutta method. At time t+1, the positions of the particles are as follows:

$$x_{t+1} = x_t + \frac{\Delta t}{6} \cdot (k_1 + 2 \cdot k_2 + 3 \cdot K_3 + k_4) \tag{8}$$

The calculation methods for k_1, k_2, k_3 and k_4 mentioned above are as follows:

$$\frac{dx}{dt} = f(t, x), \quad x(t_0) = x_0 \tag{9}$$

$$k_1 = f(t_n, x_n) \tag{10}$$

$$k_2 = f(t_n + \frac{\Delta t}{2}, x_n + \frac{\Delta t}{2} \cdot k_1) \tag{11}$$

$$k_3 = f(t_n + \frac{\Delta t}{2}, x_n + \frac{\Delta t}{2} \cdot k_2) \tag{12}$$

$$k_4 = f(t_n + \Delta t, x_n + \Delta t \cdot k3) \tag{13}$$

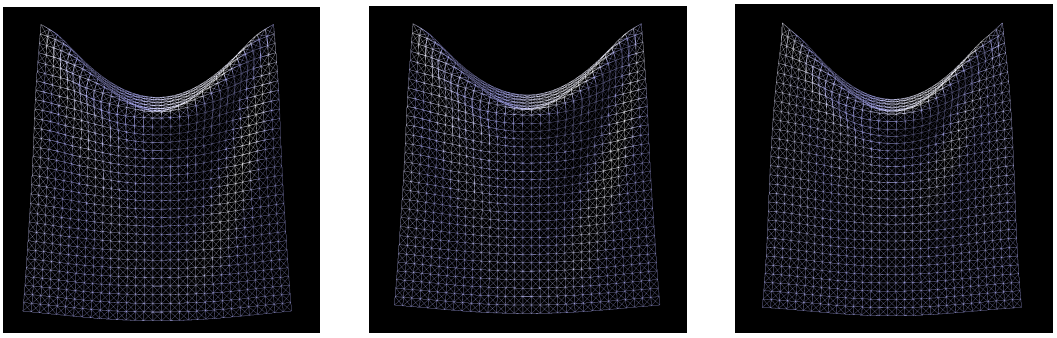
Compared to the Euler method, the fourth-order Runge-Kutta (RK4) method significantly reduces errors and improves accuracy(3).

4.5.2 Verlet Integration

The RK4 method requires four function evaluations per timestep, substantially increasing the computational load. Therefore, we must find numerical integration methods that can provide high accuracy while reducing computational demands. Verlet Integration offers significantly simpler computations with high accuracy.

$$x_{t+1} = x_t + (x_t - x_{t-1}) + a_t \cdot \Delta t^2 \tag{14}$$

where $x_t$is the current position of the mass and x_{t-1} is the position of the mass at the previous time step.



(a) Semi-Implicit Euler Method (b) Runge-Kutta Method (c) Verlet Integration Method

Figure 4: Simulation Method

The errors of the Runge-Kutta(3) and Verlet Integration(4) methods have been rigorously proven in existing paper and will not be reiterated in this project. This project compares the time required to render 2500 frames using three different methods, as shown below:

method	Semi-Implicit Euler	Runge-Kutta	Verlet Integration
time	4885511	12793878	4844902
error	O(t)	O(t^4)	O(t^2)

Table 1: The time required to refresh 2500 frames, measured in milliseconds

As shown in the table above, the rendering time for both the Verlet Integration and the Euler methods is approximately 38% of the time required by the Runge-Kutta method. However, the error for the Verlet Integration is O(t^2), and for the Euler method, it is O(t). Therefore, when a low error is required, we can consider using the Runge-Kutta method; when the error requirement is not stringent, Verlet Integration is more reasonable.

4.5.3 Collision detection and response

In our project, we implemented collision detection and response to handle cloth interactions with rigid objects (ball and cube). This process ensure the cloth react relatively realistically when it collides with these objects, maintaining physical precision.

1. **Ball**
   When a cloth mass point touches the volume of the ball, a collision reaction triggered. This collision is handles by finding the intersect point and then changing the velocity of the mass point to resolve the collision.
   Steps for ball collision response

   (a) **Distance Calculation:** For each mass, calculate the distance d from the mass point's world position P_{mass} to the ball's center c.

   $$\begin{aligned} \vec{d} &= p_{mass} - c \\ d &= \|\vec{d}\| \end{aligned} \tag{15}$$

   (b) **Penetration check:** Compute the distance d and check it is less than ball's radius r. If $d < r$, a collision will be detected.

   (c) **Solve collision:** Normalize the distance vector to get the collision normal n and compute the contact point $p_{contact}$

   $$\begin{aligned} n &= \frac{\vec{d}}{d} \\ p_{contact} &= c + n \cdot r \end{aligned} \tag{16}$$

   (d) **Repositioning and change the velocity:** Move the mass to the contact point. Reflect the mass point's velocity $v_{incoming}$ against collision normal.

   $$\begin{aligned} v_{reflected} &= v_{incoming} - 2 \cdot (v_{incoming} \cdot n)n \\ v_{mass} &= v_{reflected} \cdot f \end{aligned} \tag{17}$$

   After reflecting the velocity, we apply a friction coefficient f to simulate real-world energy loss during the collision, ensuring the cloth behave more realistically.

2. **Cube**
   The collision response with a cube using Axis-Aligned bounding Box(AABB) method involves checking if a mass is inside the cube. If so, we resolve the collision by repositioning the mass to the nearest face of the cube and reassign its velocity.
   AABB method(5)
   An AABB is a box aligned with the coordinate axes and is defined by its center and its half size in each dimention. For a cube, the half size if half the length of its edge. And the center of the AABB is the midpoint of cube.
   Steps for Cube collision Response

   (a) **Distance Calculation:** Use equation 15 to calculate the distance d from the mass point's world position P_{mass} to the cube's center c.

   (b) **Penetration check:** Check if a mass is within the bounds of the cube by comparing the absolute values of the distance vector with half size of cube $h = \frac{s}{2}$.

   $$p_x = h - |\vec{d_x}|, \quad p_y = h - |\vec{d_y}|, \quad p_z = h - |\vec{d_z}| \tag{18}$$

   If all of equations in 18 are greater than or equal to 0, the mass is inside the cube.

   (c) **Solve collision:** We solve the collision by repositioning mass position to cube's nearest face. This can be done by assigning the position vector components.

   $$nearest_face = \min(p_x, p_y, p_z) \tag{19}$$

   $$\vec{p}_{mass} = \vec{c} + \vec{d} \text{ where } \vec{d} = \begin{pmatrix} (d_x > 0)?h : -h \\ (d_y > 0)?h : -h \\ (d_z > 0)?h : -h \end{pmatrix} \tag{20}$$

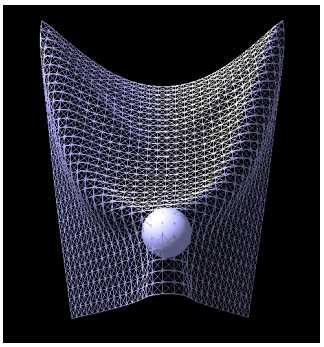
(d) **Velocity reflection:** Reflect the mass velocity $v_{incoming}$ against the normal of the nearest face.

$$n = \frac{p_{new} - p_{mass}}{\|p_{new} - p_{mass}\|} \tag{21}$$

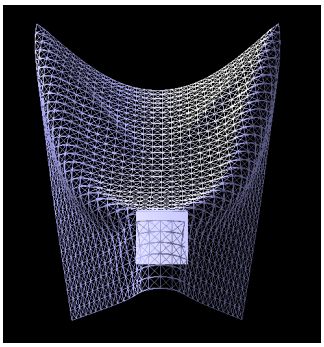
Similarly, we use equation 17 to obtain new v_{mass}.

3. **Rectangle**
   The collision response with a rectangle also uses the AABB method. Since the process is very similar to the cube collision response, we use the same steps with only slight adjustments for the rectangular dimensions. We reposition the mass by assigning components of position vector to the width, height and depth of the rectangle.

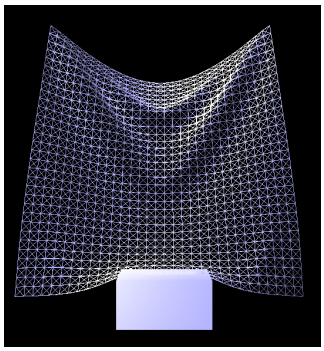
Figure 6 illustrates the result of the cloth collision with various objects. Similarly as in the real world, the mass came to a stop on the item's surface after collision. Especially, if we place the object on the edge of the cloth, the fabric will past the object and eventually stop. If we increase the friction of the object, the cloth will move more slowly.



(a) Ball collision response



(b) Cube collision response

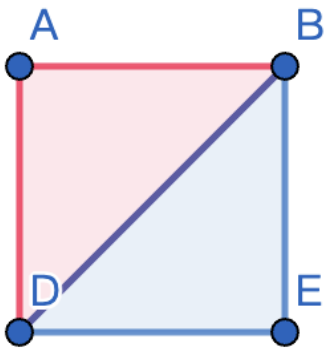


(c) Rectangle collision response

Figure 5: Deformation of cloth

4.6 Texture mapping

We divide each square block, which consists of four mass points, into two triangles. For each triangle, we apply the Blinn-Phong shading model to calculate the light intensity and use the UV coordinates to retrieve the corresponding texture data from the texture map, thus adding texture effects to the cloth.



(a) Divide triangles into cloth mesh

(b) After texture mapping. Source: `https://www.sohu.com/a/724338486_121803892#google_vignette`

Figure 6: Texture mapping

4.7 Shading

This shader uses the Phong shading model, which combines ambient and diffuse lighting with texture colour to produce the final shaded effect. The shader first obtains the final ambient lighting colour by using the light's colour and the strength of the ambient lighting. Diffuse lighting calculates the intensity of ambient lighting that varies based on the angle between the direction of the light source and the normals. Finally, it obtains the texture colour according to the texture coordinates, combining the ambient light, diffuse light, and texture colour to form the final colour.

5 Conclusion and future works

Our project enhanced the traditional mass-spring model for cloth simulation by integrating advanced techniques such as Runge-Kutta and Verlet integration, improving precision and stability. We implemented collision detection using Axis-Aligned Bounding Box (AABB) technology. Additionally, we addressed the "super-elastic" effect with deformation constraints and added dynamic wind interaction for realism. Our work significantly improves cloth simulation's accuracy and applicability in computer graphics.

5.0.1 Collision detection improvment

As for future work, we haven't solve the mould penetration in collisions. We attempted to recalculate the mass's velocity in the right direction using face normal, but the mass's location remained incorrect. Importing many obj objects and use BVH to speed up and perform collision response is another area to investigate(2).

5.0.2 Refinement

The simulated cloth does not appear soft enough when the cloth mesh is set too wide (i.e., with fewer mass points). The required computational effort significantly increases when the mesh is set too fine (i.e., with more mass points), although the simulated cloth looks more realistic. However, it is unnecessary to set all areas with a fine mesh; only the areas with folds need to be finely divided. We attempted to implement adaptive mesh refinement in this project, but we needed more resources and time constraints to implement it fully. To avoid compromising the final report, we had to remove this feature. After team discussions and research, refining rectangular meshes is relatively tricky, while refining triangular meshes is comparatively more straightforward. If we had more time, we would attempt to modify the mesh into triangular shapes and implement the refinement feature.

6 Acknowledgement

We would like to thanks the developers and maintainers of the following resources that significantly contributed to our project:

- The LearnOpen GL repository by Joey De Vries, which provides many examples that helps us to implement OpenGl in our project.
- The ClothSimulation repository by xxMeow, which shows examples on how to use GLM for cloth simulation.

References

[1] Provot, X. (1995) Deformation Constraints in a Mass-Spring Model to Describe Rigid Cloth Behaviour. In *Proceedings of Graphics Interface '95*, pp. 147–154.

[2] Wang, X., Tang, M., Manocha, D. and Tong, R. (2018), Efficient BVH-based Collision Detection Scheme with Ordering and Restructuring. In *Computer Graphics Forum, '37'*, pp. 227-237.

[3] J. C. Butcher, (2008) *Numerical Methods for Ordinary Differential Equations*, 2nd edition, John Wiley & Sons

[4] Loup Verlet (1967), "Computer 'Experiments' on Classical Fluids. I. Thermodynamical Properties of Lennard-Jones Molecules," *Physical Review*.

[5] Ericson, Christer. (2005) *Real-Time Collision Detection*. Morgan Kaufmann Publishers, pp. 77–87.