# Deep Contact

## Accelerating Rigid Simulation with Convolutional Networks

J. Wu

Department of Computer Science
University of Copenhagen

Master Thesis Defense, 2018

# Outline

# Outline

# Previous Work

- Deep Learning applied in computer simulation.
- Speed up contact simulation.

# Outline

1. Introduction
   - Previous Work
   - Thesis Overview
2. Particles-Grid-Particles
   - Grid-Particle Method
   - Smoothed Particle Hydrodynamics
   - Bilinear Interpolation
3. Deep Learning Model
   - Convolutional Neural Networks
   - CNN Architecture
   - Training Configuration
4. Results and Analysis
5. Future Work

# Thesis Overview

# Thesis Overview

**Model Contact**

# Thesis Overview

**Model Contact**

$$\boldsymbol{M}\dot{\mathbf{u}} = \underbrace{\boldsymbol{J}^T\boldsymbol{\lambda}}_{\mathbf{F}_{contact}} + \mathbf{F}_{ext}$$

# Thesis Overview

**Model Contact**

$$\boldsymbol{M}\dot{\mathbf{u}} = \underbrace{\boldsymbol{J}^T\boldsymbol{\lambda}}_{\mathbf{F}_{contact}} + \mathbf{F}_{ext}$$

Then, it can be transformed to a Linear Complementarity Problem

**Model Contact**

$$\boldsymbol{M}\dot{\mathbf{u}} = \underbrace{\boldsymbol{J}^T\boldsymbol{\lambda}}_{\mathbf{F}_{contact}} + \mathbf{F}_{ext}$$

Then, it can be transformed to a Linear Complementarity Problem

$$\begin{aligned}
\mathbf{w} &= \boldsymbol{J}_n\mathbf{v}^{t+1} \\
&= \underbrace{\boldsymbol{J}_n\boldsymbol{M}^{-1}\boldsymbol{J}_n^T\Delta t}_{\boldsymbol{A}}\boldsymbol{\lambda}_n + \underbrace{\boldsymbol{J}_n(\Delta t\boldsymbol{M}^{-1}\mathbf{F}_{ext} + \mathbf{v}^t)}_{\boldsymbol{b}} \\
&= \boldsymbol{A}\boldsymbol{\lambda}_n + \boldsymbol{b}
\end{aligned}$$

# Thesis Overview

**Model Contact**

$$\boldsymbol{M}\dot{\mathbf{u}} = \underbrace{\boldsymbol{J}^T\boldsymbol{\lambda}}_{\mathbf{F}_{contact}} + \mathbf{F}_{ext}$$

Then, it can be transformed to a Linear Complementarity Problem

$$\begin{aligned}\mathbf{w} &= \boldsymbol{J}_n\mathbf{v}^{t+1} \\ &= \underbrace{\boldsymbol{J}_n\boldsymbol{M}^{-1}\boldsymbol{J}_n^T\Delta t}_{\boldsymbol{A}}\boldsymbol{\lambda}_n + \underbrace{\boldsymbol{J}_n(\Delta t\boldsymbol{M}^{-1}\mathbf{F}_{ext} + \mathbf{v}^t)}_{\boldsymbol{b}} \\ &= \boldsymbol{A}\boldsymbol{\lambda}_n + \boldsymbol{b}\end{aligned}$$

$$\boldsymbol{\lambda} = LCP(\boldsymbol{A}, \boldsymbol{b})$$

**Data:** $N, \boldsymbol{\lambda}_{init}, \boldsymbol{A}, \boldsymbol{b}$
**Result:** Compute the values of $\boldsymbol{\lambda}$, the convergence rate $\boldsymbol{\theta}$

```
for k = 1 To N do
    if k = 1 then
        λ ← λ_init
    end
    λ_old ← λ
    for all i do
        r_i ← A_{i*} λ + b_i ;
        λ_i ← max(0, λ_i − r_i / A_ii) ;
    end
    θ_k ← max(|λ − λ_old|)
end
```

$$\boldsymbol{r}_i \leftarrow \boldsymbol{A}_{i*}\boldsymbol{\lambda} + \boldsymbol{b}_i \qquad \boldsymbol{\lambda}_i \leftarrow \max\left(0, \boldsymbol{\lambda}_i - \frac{\boldsymbol{r}_i}{\boldsymbol{A}_{ii}}\right) \qquad \boldsymbol{\theta}_k \leftarrow \max(|\boldsymbol{\lambda} - \boldsymbol{\lambda}_{old}|)$$

# Outline

# Grid-Particle Method

In order to generate accessible data for CNN model, we transform every state into a set of grid images.

- It can make the simulation states be expressed by a set of matrixes, which can be accessible for deep neural networks.

# Grid-Particle Method

In order to generate accessible data for CNN model, we transform every state into a set of grid images.

- It can make the simulation states be expressed by a set of matrixes, which can be accessible for deep neural networks.
-

# Grid-Particle Method

## Workflow

The whole workflow can be described as,

1. Based on Smoothed Particle Hydrodynamics(SPH), map current state($m, v_x, v_y, \omega, n_x$) to a image(the number of channel is 5.), which is called feature image.

The whole workflow can be described as,

1. Based on Smoothed Particle Hydrodynamics(SPH), map current state($m, v_x, v_y, \omega, n_x$) to a image(the number of channel is 5.), which is called feature image.
2. The feature image will be used as input to a model(created by a convolutional neural network), then one image(the number of channels is 2) will be getting, which can be called label image.

The whole workflow can be described as,

1. Based on Smoothed Particle Hydrodynamics(SPH), map current state($m, v_x, v_y, \omega, n_x$) to a image(the number of channel is 5.), which is called feature image.
2. The feature image will be used as input to a model(created by a convolutional neural network), then one image(the number of channels is 2) will be getting, which can be called label image.
3. For all contacts positions, interpolated values will be gener- ated based on label image. Then, the values will be used as starting iterate values for contact force solver. In our hypoth- esis, the given starting values will speed up the solver to reach convergence.

# Outline

The heart of SPH is a kernel interpolation method which allows any function to be expressed in terms of its values at a set of disordered points - the particles.

# Smoothed Particle Hydrodynamics

## Fundamentals

The heart of SPH is a kernel interpolation method which allows any function to be expressed in terms of its values at a set of disordered points - the particles.

$$A_I(\mathbf{r}) = \int A(\mathbf{r}')W(\|\mathbf{r} - \mathbf{r}'\|, h)\mathrm{d}\mathbf{r}' \tag{1}$$

The heart of SPH is a kernel interpolation method which allows any function to be expressed in terms of its values at a set of disordered points - the particles.

$$A_I(\mathbf{r}) = \int A(\mathbf{r}')W(\|\mathbf{r} - \mathbf{r}'\|, h)\mathrm{d}\mathbf{r}' \tag{1}$$

where the integration is over the entire space, and $W$ is an in- terpolating kernel with

The heart of SPH is a kernel interpolation method which allows any function to be expressed in terms of its values at a set of disordered points - the particles.

$$A_I(\mathbf{r}) = \int A(\mathbf{r}')W(\|\mathbf{r} - \mathbf{r}'\|, h)\mathrm{d}\mathbf{r}' \tag{1}$$

where the integration is over the entire space, and $W$ is an in- terpolating kernel with

$$\int W(\|\mathbf{r} - \mathbf{r}'\|, h)\mathrm{d}\mathbf{r}' = 1 \tag{2}$$

For numerical work,

The heart of SPH is a kernel interpolation method which allows any function to be expressed in terms of its values at a set of disordered points - the particles.

$$A_I(\mathbf{r}) = \int A(\mathbf{r}')W(\|\mathbf{r} - \mathbf{r}'\|, h)\mathrm{d}\mathbf{r}' \tag{1}$$

where the integration is over the entire space, and $W$ is an in- terpolating kernel with

$$\int W(\|\mathbf{r} - \mathbf{r}'\|, h)\mathrm{d}\mathbf{r}' = 1 \tag{2}$$

For numerical work,

$$A_S(\mathbf{x}) = \sum_i A(\mathbf{x}_i)W(\|\mathbf{x}_i - \mathbf{x}\|, h) \tag{3}$$

- **Poly6**

- **Poly6**

$$W_{poly6}(\mathbf{r}, h) = \frac{315}{64\pi h^9} \begin{cases} (h^2 - \|\mathbf{r}\|^2)^3 & 0 \leq \|\mathbf{r}\| \leq h \\ 0 & \text{Otherwise} \end{cases} \tag{4}$$

- **Poly6**

$$W_{poly6}(\mathbf{r}, h) = \frac{315}{64\pi h^9} \begin{cases} (h^2 - \|\mathbf{r}\|^2)^3 & 0 \le \|\mathbf{r}\| \le h \\ 0 & \text{Otherwise} \end{cases} \quad (4)$$

- **Spicky**

- **Poly6**

$$W_{poly6}(\mathbf{r}, h) = \frac{315}{64\pi h^9} \begin{cases} (h^2 - \|\mathbf{r}\|^2)^3 & 0 \leq \|\mathbf{r}\| \leq h \\ 0 & \text{Otherwise} \end{cases} \tag{4}$$

- **Spicky**

$$W_{spiky}(\mathbf{r}, h) = \frac{15}{\pi h^6} \begin{cases} (h - \|\mathbf{r}\|)^3 & 0 \leq \|\mathbf{r}\| \leq h \\ 0 & \text{Otherwise} \end{cases} \tag{5}$$

# Smoothed Particle Hydrodynamics

## Kernels

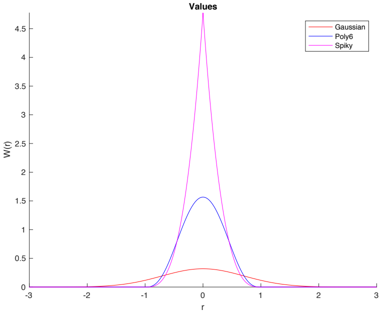



Figure: Comparation of different kernels, we set smoothing length $h = 1$ here.

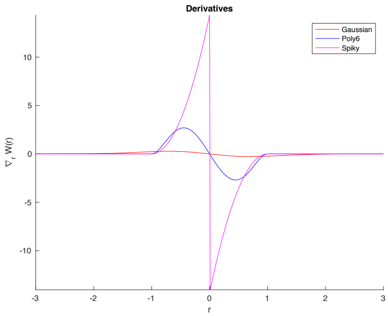



Figure: Comparation of gradient of different kernels, we set $h = 1$ here.

# Mapping bodies into a grid image

**Data:** Given a set of bodies $\mathcal{B}$ and the state in time $t$, as well as all the spacial positions of grid nodes $\mathbf{x}$

**Result:** the body grid image $\boldsymbol{G}_{\mathcal{B}}$

**for** *all i and j* **do**

1. Find the nearest neihbors $\mathcal{B}_{near} \subseteq \mathcal{B}$ around $\mathbf{x}_{ij}$
2. read the current state,

$$m_k, \boldsymbol{v}_k, \boldsymbol{q}_k, \omega_k \quad k \in \mathcal{B}_{near}$$

3. Define state vector $\boldsymbol{S}_k \leftarrow [m_k, v_{kx}, v_{ky}, \omega_k]$
4. Compute grid values

$$\boldsymbol{G}_{\mathcal{B}}(i,j) \leftarrow \sum_{k \in \mathcal{B}_{near}} W(\mathbf{x}_{ij}, \boldsymbol{q}_k)\boldsymbol{S}_k$$

**end**

# Mapping contacts into a grid image

**Data:** Given a set of contacts $\mathcal{C}$ between a set of bodies $\mathcal{B}$ and the state in time $t$, as well as all the spacial positions of grid nodes $\mathbf{x}$

**Result:** the contact grid image $\boldsymbol{G}_\lambda$

**for** *all i and j* **do**

1. Find the nearest neihbors $\mathcal{C}_{near} \subseteq \mathcal{C}$ around $\mathbf{x}_{ij}$
2. read the current contact forces values and its position.

$$\boldsymbol{q}_k, \boldsymbol{\lambda}_k \quad k \in \mathcal{C}_{near} \quad \boldsymbol{\lambda} = [\lambda_n, \lambda_t]$$

3. Compute grid values

$$\boldsymbol{G}_\lambda(i,j) \leftarrow \sum_{k \in \mathcal{C}_{near}} W(\mathbf{x}_{ij}, \boldsymbol{q}_k)\boldsymbol{\lambda}_k$$

**end**

# Outline

# Bilinear Interpolation

Once getting contact grid image, we need transform the grid image to a physical state, specific contact values in each contact point. We applied bilinear interpolation in our case.

# Bilinear Interpolation

Once getting contact grid image, we need transform the grid image to a physical state, specific contact values in each contact point. We applied bilinear interpolation in our case.

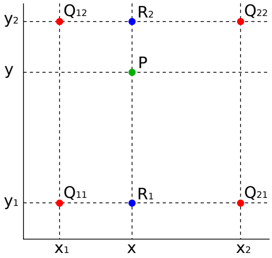



Figure: The figure shows the visualization of bilinear interpolation. The four red dots show the data points and the green dot is the point at which we want to interpolate.

# Bilinear

we can firstly do linear interpolation in the $x$-direction. This yields

# Bilinear

we can firstly do linear interpolation in the $x$-direction. This yields

$$f(x, y_1) \approx \frac{x_2 - x}{x_2 - x_1} f(Q_{11}) + \frac{x - x_1}{x_2 - x_1} f(Q_{21}), \tag{6a}$$

$$f(x, y_2) \approx \frac{x_2 - x}{x_2 - x_1} f(Q_{12}) + \frac{x - x_1}{x_2 - x_1} f(Q_{22}). \tag{6b}$$

# Bilinear

we can firstly do linear interpolation in the $x$-direction. This yields

$$f(x, y_1) \approx \frac{x_2 - x}{x_2 - x_1} f(Q_{11}) + \frac{x - x_1}{x_2 - x_1} f(Q_{21}), \tag{6a}$$

$$f(x, y_2) \approx \frac{x_2 - x}{x_2 - x_1} f(Q_{12}) + \frac{x - x_1}{x_2 - x_1} f(Q_{22}). \tag{6b}$$

After getting the two values in $x-$direction $f(x, y_1)$ and $f(x, y_2)$, we can combine these values to do interpolation in $y-$ direction.

# Bilinear

we can firstly do linear interpolation in the $x$-direction. This yields

$$f(x, y_1) \approx \frac{x_2 - x}{x_2 - x_1} f(Q_{11}) + \frac{x - x_1}{x_2 - x_1} f(Q_{21}), \tag{6a}$$

$$f(x, y_2) \approx \frac{x_2 - x}{x_2 - x_1} f(Q_{12}) + \frac{x - x_1}{x_2 - x_1} f(Q_{22}). \tag{6b}$$

After getting the two values in $x-$direction $f(x, y_1)$ and $f(x, y_2)$, we can combine these values to do interpolation in $y-$ direction.

$$f(x, y) \approx \frac{y_2 - y}{y_2 - y_1} f(x, y_1) + \frac{y - y_1}{y_2 - y_1} f(x, y_2) \tag{7}$$

# SPH method
## Experiments

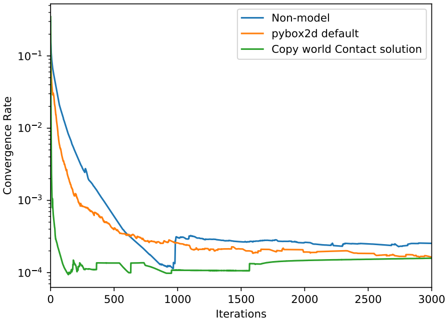



Figure: Average convergence rate for different models(not including **SPH-based model**).

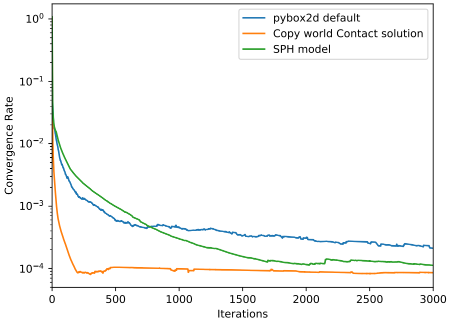
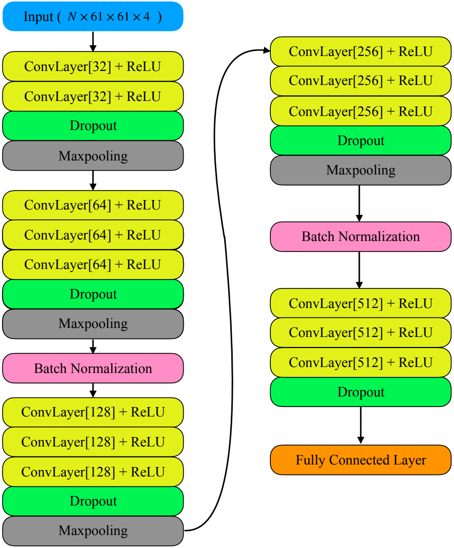



Figure: Average convergence rate for different models(not including **SPH-based model**).

# Outline

# Outline

# CNN Architecture



# Outline

# Training Hyperparameter

| Hyperparameter | Setting |
| --- | --- |
| Activation function | ReLU |
| Weight initilization | He normal |
| Weight regularizer | L2 |
| Convolution border mode | Same |
| Stride | 2 |
| Kernel size | $(3, 3)$ |
| Dropout rate | 0.1 |
| Optimizer | SGD |
| Initial Learning Rate | $1 \times 5 \times 10^{-3}$ |
| Batch Size | 200 |
| Epoch | 1000 |
| Validation Rate | 0.2 |

Table: Hyperparameter settings.

# Simulation Configuration

- **World Setting**

# Simulation Configuration

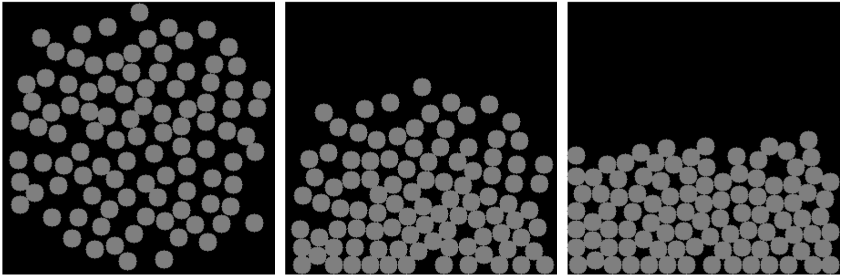
- **World Setting** the world box size is $30 \times 30$, and there are $50 - 100$ circle rigid bodies($r = 1$, all circle rigid bodies in the same size.) inside the box. Initially, the rigid circles will be located following gaussian distribution2. Then, all rigid circles will fall down by gravity.

# Simulation Configuration

- **World Setting** the world box size is $30 \times 30$, and there are $50 - 100$ circle rigid bodies($r = 1$, all circle rigid bodies in the same size.) inside the box. Initially, the rigid circles will be located following gaussian distribution2. Then, all rigid circles will fall down by gravity.
- **Simulation Setting**

# Simulation Configuration

- **World Setting** the world box size is $30 \times 30$, and there are $50 - 100$ circle rigid bodies($r = 1$, all circle rigid bodies in the same size.) inside the box. Initially, the rigid circles will be located following gaussian distribution2. Then, all rigid circles will fall down by gravity.
- **Simulation Setting** there will be totally 600-steps simulation. For each step, $\Delta t = 0.01s$, and the number of iteration in each step will be set as fixed, 3000.



(a) Time Step=0 (b) Time Step=200 (c) Time Step=400

Figure: Visualization for experiment simulation

I define $\boldsymbol{d} = (d_x, d_y)$ as grid cell size and $h$ as smoothing length.

I define $\boldsymbol{d} = (d_x, d_y)$ as grid cell size and $h$ as smoothing length.

- Since the objects are circles, $d_x = d_y = d$

I define $\boldsymbol{d} = (d_x, d_y)$ as grid cell size and $h$ as smoothing length.

- Since the objects are circles, $d_x = d_y = d$
- $d$ must be less than the distance of nearest two contact points. It can be defined. $d \leq r = 1$

I define $\boldsymbol{d} = (d_x, d_y)$ as grid cell size and $h$ as smoothing length.

- Since the objects are circles, $d_x = d_y = d$
- $d$ must be less than the distance of nearest two contact points. It can be defined. $d \leq r = 1$
- the smooth length $h$ should be less than the minimum distance between two contact points $d$, $h \leq d$

I define $\boldsymbol{d} = (d_x, d_y)$ as grid cell size and $h$ as smoothing length.

- Since the objects are circles, $d_x = d_y = d$
- $d$ must be less than the distance of nearest two contact points. It can be defined. $d \leq r = 1$
- the smooth length $h$ should be less than the minimum distance between two contact points $d$, $h \leq d$
- For a given $d$, $h \geq \frac{\sqrt{2}}{2} d \approx 0.71d$

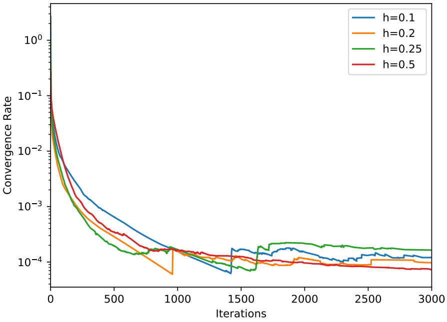



Figure: The grid size $d$ is set 0.25. $h = 0.1, 0.2, 0.25, 0.5$ is tested respectively. This figure shows different coveragence rate based on different $h$ value. The kernel is **Poly6**

# Results and Analysis

## SPH parameters(**Poly6**)

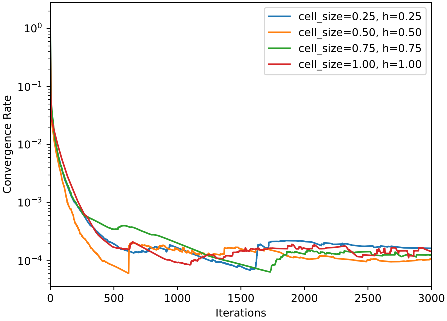



Figure: Coveragence rate for different *d* value. The kernel is **Poly6**

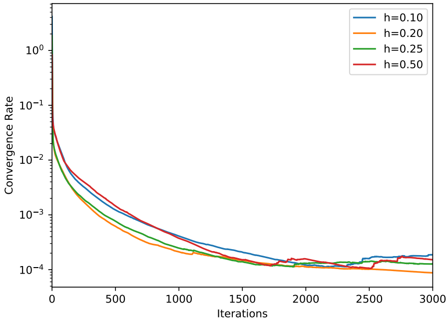



Figure: The grid size $d$ is set 0.25. $h = 0.1, 0.2, 0.25, 0.5$ is tested respectively. This figure shows different coveragence rate based on different $h$ value. The kernel is **Spiky**

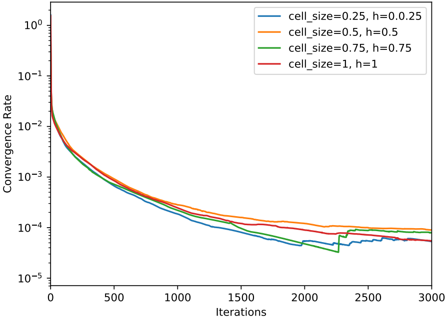



Figure: Coveragence rate for different $d$ value. The kernel is **Spiky**

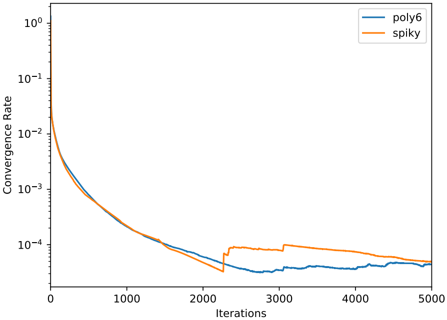



Figure: Coveragence rate for kernel **Poly6** anf **Spiky**. $h_{\text{poly6}} = d_{\text{poly6}} = 0.5$, while $h_{\text{spiky}} = d_{\text{spiky}} = 0.25$

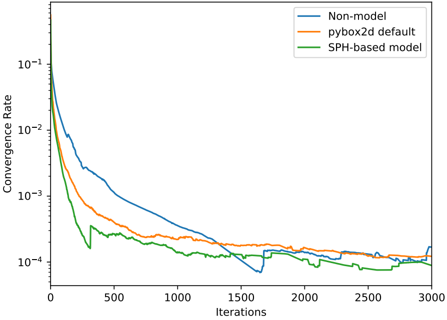



Figure: Coveragence rate for models(different initial values for $\boldsymbol{\lambda}$).

- **Input Size**, the input will be $61 \times 61 \times 4$. Since the orginal world is $30 \times 30$ and grid size is $d = 0.5$, the generated grid would be $61 \times 61$. There would 4 channels$[m, v_x, v_y, \omega]$

- **Input Size**, the input will be $61 \times 61 \times 4$. Since the orginal world is $30 \times 30$ and grid size is $d = 0.5$, the generated grid would be $61 \times 61$. There would 4 channels$[m, v_x, v_y, \omega]$
- **Output Size**, output size depends on the label size. The orginal label image would be $[\lambda_n, \lambda_t]$, so the label image size would be $61 \times 61 \times 2$, which should be flattened as the actual training label. The label size would be $61 \times 61 \times 2$

- **Input Size**, the input will be $61 \times 61 \times 4$. Since the orginal world is $30 \times 30$ and grid size is $d = 0.5$, the generated grid would be $61 \times 61$. There would 4 channels$[m, v_x, v_y, \omega]$
- **Output Size**, output size depends on the label size. The orginal label image would be $[\lambda_n, \lambda_t]$, so the label image size would be $61 \times 61 \times 2$, which should be flattened as the actual training label. The label size would be $61 \times 61 \times 2$
- **Weights Number**, the total weights number is $64,498,866$.

- **Input Size**, the input will be $61 \times 61 \times 4$. Since the orginal world is $30 \times 30$ and grid size is $d = 0.5$, the generated grid would be $61 \times 61$. There would 4 channels$[m, v_x, v_y, \omega]$
- **Output Size**, output size depends on the label size. The orginal label image would be $[\lambda_n, \lambda_t]$, so the label image size would be $61 \times 61 \times 2$, which should be flattened as the actual training label. The label size would be $61 \times 61 \times 2$
- **Weights Number**, the total weights number is $64,498,866$.
- **Training Environment**, GPU(*GeForce GTX 1080 Ti, 11 Gbps GDDR5X memory*) held by Image Section, DIKU.

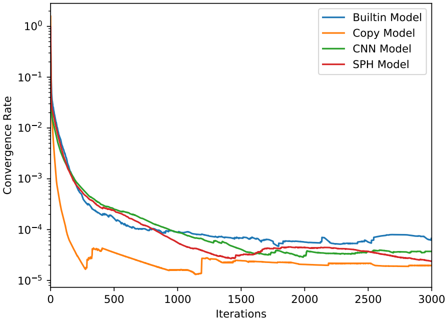



Figure: The final result. Add the final CNN solution to compare with other methods.

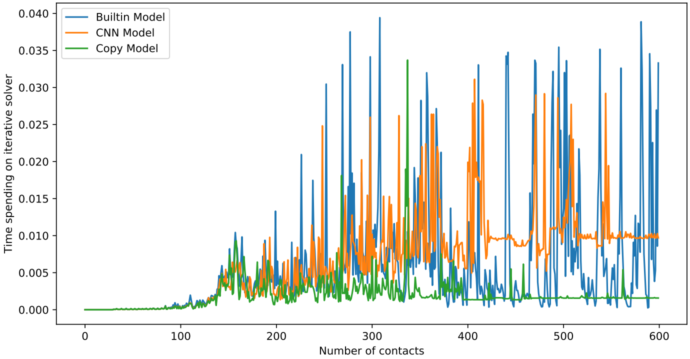



Figure: Time spent for contact solver iteration

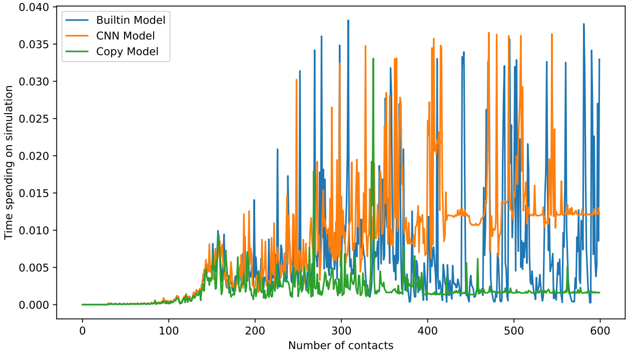



Figure: Time spend for the whole contact solution(including warm starting calculation).

Now, we can obtain some conclusions,

Now, we can obtain some conclusions,

- **CNN-Model** actually makes the iterative solver converge faster. But it is not a great improvement to buit-in warm starting.

Now, we can obtain some conclusions,

- **CNN-Model** actually makes the iterative solver converge faster. But it is not a great improvement to buit-in warm starting.
- **CNN-Model** performs very similar to **SPH-Model**, which means the CNN predicts contact image well.

Now, we can obtain some conclusions,

- **CNN-Model** actually makes the iterative solver converge faster. But it is not a great improvement to buit-in warm starting.
- **CNN-Model** performs very similar to **SPH-Model**, which means the CNN predicts contact image well.
- Due to the limitation of the SPH-based method, **CNN-Model** just gets a small improvement compared with **Builtin-Model**. And it cannot perform as good as **Copy-Model**.

# Future Work

1. **Grid-Particles Method**

# Future Work

1. **Grid-Particles Method**
    - **SPH**

# Future Work

1. **Grid-Particles Method**
   - **SPH** For the SPH-based method, it is still far away from perfect. It performs not much better than warm starting. The probable attempt will including trying more new kernels, using new data and algorithm for nearest neighbor searching.

# Future Work

1. **Grid-Particles Method**
   - **SPH** For the SPH-based method, it is still far away from perfect. It performs not much better than warm starting. The probable attempt will including trying more new kernels, using new data and algorithm for nearest neighbor searching.
   - **Interpolation Method**

# Future Work

1. **Grid-Particles Method**
    - **SPH** For the SPH-based method, it is still far away from perfect. It performs not much better than warm starting. The probable attempt will including trying more new kernels, using new data and algorithm for nearest neighbor searching.
    - **Interpolation Method** It might lose some essential informa- tion when it was interpolated back to particles. So exploring another interpolation method would be helpful to this project.

# Future Work

1. **Grid-Particles Method**
   - **SPH** For the SPH-based method, it is still far away from perfect. It performs not much better than warm starting. The probable attempt will including trying more new kernels, using new data and algorithm for nearest neighbor searching.
   - **Interpolation Method** It might lose some essential informa- tion when it was interpolated back to particles. So exploring another interpolation method would be helpful to this project.
2. **Deep Learning Model**,

# Future Work

1. **Grid-Particles Method**
   - **SPH** For the SPH-based method, it is still far away from perfect. It performs not much better than warm starting. The probable attempt will including trying more new kernels, using new data and algorithm for nearest neighbor searching.
   - **Interpolation Method** It might lose some essential informa- tion when it was interpolated back to particles. So exploring another interpolation method would be helpful to this project.
2. **Deep Learning Model**, More learning models can be explored, like Recurrent Neural Networks(RNN), Long short-term memory(LSTM).

# Future Work

1. **Grid-Particles Method**
   - **SPH** For the SPH-based method, it is still far away from perfect. It performs not much better than warm starting. The probable attempt will including trying more new kernels, using new data and algorithm for nearest neighbor searching.
   - **Interpolation Method** It might lose some essential informa- tion when it was interpolated back to particles. So exploring another interpolation method would be helpful to this project.
2. **Deep Learning Model**, More learning models can be explored, like Recurrent Neural Networks(RNN), Long short-term memory(LSTM).
3. **More Shapes Experiments**

# For Further Reading I


A. Author.
*Handbook of Everything*.
Some Press, 1990.

S. Someone.
On this and that.
*Journal of This and That*, 2(1):50–100, 2000.