Learning Extremely High Density Crowds as Active Matters Supplemental Material

Feixiang He[2], Jiangbei Yue[3], Jialin Zhu[2], Armin Seyfried[4], Dan Casas[5]*, Julien Pettré[6], He Wang [1,2†]
[1]AI Centre, University College London, UK [2]University College London, UK
[3]University of Leeds, UK [4] Forschungszentrum Jülich, Germany
[5]Universidad Rey Juan Carlos, Spain [6] INRIA Rennes, France

1. Additional Technical Details

1.1. Loss function

During training, We employ the Mean Square Error loss between the predicted velocity field $\{\hat{\mathbf{v}}^t\}_{t=1}^T$ and the ground truth $\{\mathbf{v}^t\}_{t=0}^T$ to optimize the model:

$$\min \frac{1}{T}\int_{t=1}^{T} ||\mathbf{v}^t - \hat{\mathbf{v}}^t||_2^2 \approx \frac{1}{T}\sum_{t=1}^{T} ||\mathbf{v}^t - \hat{\mathbf{v}}^t||_2^2 \quad (1)$$

where we can only approximate the first integral as the summation over observed frames. But the nature of the approximation does not dictate how the frames are distributed on the timeline. They can be evenly or unevenly distributed.

1.2. Error Metrics

To evaluate our proposed model, we employ two metrics: Err_{vel} and Err_{flow}. As presented in Sec. 4 of the main paper, Err_{vel} is used to assess the velocity field locally and is calculated as:

$$Err_{vel} = \frac{1}{T}\sum_{t=1}^{T} ||\mathbf{v}^t - \hat{\mathbf{v}}^t||_2^2 \quad (2)$$

Differently, Err_{flow} operates directly on the raw optical flow data and is defined as:

$$Err_{flow} = \frac{1}{T}\sum_{t=1}^{T} ||\mathbf{o}^t - \hat{\mathbf{o}}^t||_2^2 \quad (3)$$

Here, $\{\hat{\mathbf{o}}^t\}_{t=1}^T$ and $\{\mathbf{o}^t\}_{t=0}^T$ represent the predicted optical flow and the ground truth respectively. Note that our model predicts the velocity field instead of directly predicting optical flow. Therefore, the predicted velocity field is converted into optical flow via G2P transfer, as detailed in sec.3.2 of the main paper.

*work done prior to joining Amazon
†Corresponding author, he_wang@ucl.ac.uk

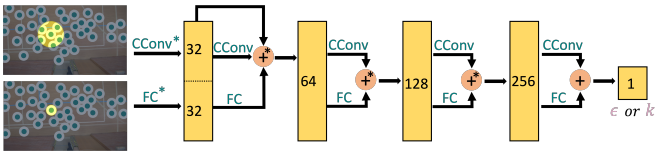

Figure 1. Parameter estimation network. Operations annotated by * are followed by a ReLU activation.

1.3. Neural Networks for Crowd Material

As illustrated in Sec 3.6 in the main paper, there are two types of learnable parameters: Young's modulus ϵ and k. These parameters are estimated using neural networks, specifically NN_ϵ for ϵ and NN_k for k. Both networks share the same architecture, as shown in Fig. 1. The design is mainly to consider the motions of particles and its neighboring particles. Since we model a continumm where particles can be anywhere and not necessarily evenly spaced, one of the component in the network is the Continuous Convolution or CCov, inspired by the convolution layer designed originally for a Lagrangian method [11]. The continuous convolution captures the relative motions of the particles in the neighborhood, which dictates the local deformation of the field. In addition, we assume that the material parameter is different for every particle (pedestrian) in space and time. Therefore, we also use a fully-connected component (FC) to allow the individual particle motion to play a bigger role compared with its neighbors whose motions are already considered in the continuous convolution of the neighborhood. After several repeated layers of CCov and FC components, the network finally outputs the target parameter.

1.4. Conditional Variational Autoencoder for Random Active Forces

As outlined in the main paper, active force learning is divided into two parts. The first part, motion alignment (represented by $\alpha\mathbf{v}$), is learned through a straightforward 6-layer convolutional neural network NN_α , where the kernel

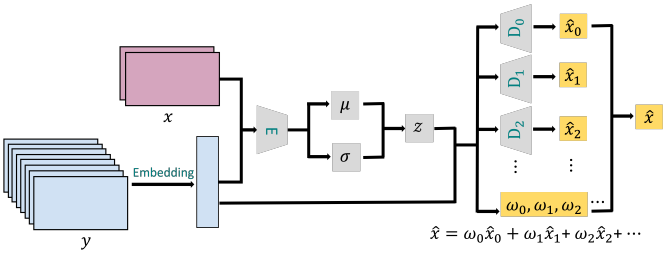

Figure 2. Architecture of CVAE.

size, stride, and padding are set to 3, 1, and 1, respectively. Each layer, except the final one, is followed by a Tanh activation function. The channel configuration for the layers is 32, 64, 128, 64, 32, and 1, respectively. The input of NN_α is the velocity field at the current step, and its output is the corresponding α.

Secondly, to model the rest (expressed as $-\beta|\mathbf{v}|^2\mathbf{v} + D_L\nabla(\nabla\cdot\mathbf{v}) + D_1\nabla^2\mathbf{v} + D_2(\mathbf{v}\cdot\nabla)^2\mathbf{v} + \tilde{\mathbf{f}}$), a conditional variational autoencoder is utilized, as illustrated in Fig. 2. The CVAE takes two inputs: x, representing the ground truth of the remaining active forces, and y, serving as the condition. The condition is composed of four terms $|\mathbf{v}|^2\mathbf{v}$, $\nabla(\nabla\cdot\mathbf{v})$, $\nabla^2\mathbf{v}$ and $(\mathbf{v}\cdot\nabla)^2\mathbf{v}$, and processed through an embedding layer. Note that the decoder consists of multiple components $(D_0, D_1, D_2, \cdots)$ which generate intermediate outputs $(\hat{x}_0, \hat{x}_1, \hat{x}_2, \cdots)$. These intermediate results are combined using a weighted sum with weights $(\omega_0, \omega_1, \omega_2, \cdots)$ to produce the final output $\hat{x}$. This architecture allows the CVAE to effectively model the complex relationships between the input and the conditional data, making it suitable for learning the intricate dynamics.

1.5. Optical flows to/from grid

Optical flows can be transferred to and from a grid representation to for training, testing, analysis and comparison. When transferring optical flow data to a grid, we make use of the particle-to-grid operation in the MPM (P2G). Each pixel in the optical flow field is treated as an individual particle, with its velocity mapped to the corresponding grid cell using a B-spline function. Conversely, when transforming the velocity field from a grid back to its corresponding optical flows, we use the grid-to-particle operation in the MPM (G2P). G2P computes the velocity at grid nodes back to each pixel, which is treated as the optical flow at that pixel.

This P2G then G2P mapping between optical flows and the velocity field, aside from facilitate learning, has a smoothening effect due to the B-spline based interpolation, effectively removing random noises from the noisy optical flows. In addition, this mapping also preserve the spatial and temporal dynamics. This dual representation enables the comparisons of the particle-level details and the grid-level velocity.

1.6. Hyperparameters

In all datasets, the density is fixed at 1. This is because it is not possible estimate each individual's mass, and if it were, there is no commonly agreed way of defining the density in high-density crowds [6]. By defining the density to be 1, the particle mass depends on the particle radius. The particle radii are set to 20 pixels on $Drill_{1-3}$, 5 pixels on $Hajj$ and $Hellfest$, 10 pixels on $Marathon$. Note this setting will not affect the learning of the dynamics as the dynamics are learned mainly through the material parameters and the active force.

For training, we employ Adam optimizer with an initial learning rate of 1×10^{-4}. To dynamically adjust the learning rate during training, we apply a LambdaLR scheduler with a decay function defined as $lr = 1 \times 10^{-4} \times 0.9^{epoch/50}$, epoch is the current iteration. The batch size is set to 4.

1.7. Adaptation of baseline methods

To demonstrate the superiority of out method, we adapt several closest methods as baselines. **Baseline**$_I$ is a physics-based method that models crowds using a weakly incompressible fluid framework [7]. PredFlow [5] is a deep learning method designed for predicting future scene parsing and motion dynamics. Additionally, we include several sequence prediction models, PredRNNv2 [12], SimVP [1], and TAU [8], which are also pure deep learning approaches and intended for video prediction tasks. PredFlow takes four consecutive RGB frames as input and outputs the future optical flows in an autoregressive fashion, **Baseline**$_I$, however, simulates velocity fields but we need to exhaustively tune the parameters. For the video prediction methods, we adapt them to take optical flow as input and forecast future optical flow sequences. Specifically, during the training, we take $1.2s$ optical flow as input and forecast the next $1.2s$ optical flow. These baselines span a diverse range of architectures, from physics-based models to cutting-edge deep learning frameworks, ensuring comprehensive coverage across methodologies.

1.8. Simulation

Our simulation runs $\sim$ 40 FPS for 75 individuals. Like any simulator, our method experiences a decrease in performance when scaling to a large number of agents. Also, albeit now shown in the paper, our method allows dynamic adding or removing objects (dynamic or static), environmental changes or events (*e.g.* explosion, evacuation). The first two can be easily handled by boundary conditions, while the latter can be realized by adding particle forces to repel people from certain places, *e.g.* explosion, or attract them, *e.g.* evacuation.

Percent	Err_{vel}							Err_{flow}						
	$1.2s$	$2.4s$	$3.6s$	$4.8s$	$6.0s$	$7.2s$	$8.4s$	$1.2s$	$2.4s$	$3.6s$	$4.8s$	$6.0s$	$7.2s$	$8.4s$
0%	1.1099	1.1748	1.1600	1.1332	1.1128	1.0954	1.0721	1.8016	1.8463	1.8072	1.7574	1.7185	1.6883	1.6558
50%	1.1078	1.1632	1.1419	1.1155	1.0966	1.0850	1.0640	1.7958	1.8303	1.7843	1.7349	1.6974	1.6729	1.6435
70%	1.1121	1.1637	1.1381	1.1069	1.0853	1.0705	1.0729	1.8086	1.8386	1.7891	1.7339	1.6942	1.6653	1.6378

Table 1. Evaluation on temporal discontinuities on $Drill_2$.

2. Additional Experimental Results

2.1. Continuous-time Prediction

As aforementioned in the main paper, one distinctive feature of our method is that it is a continuous-time model, which does not require data to be observed on evenly discretized timeline. This is because the core of our method is a PDE which can be solved for any horizon in future. This feature provides good flexibility in terms of data collection. For CCTV camera videos, sometimes there can be objects *e.g.* tree leaves flying across the view or certain sudden change of the lighting condition. Under these conditions, the optical flows will be extremely unreliable. This requires certain data pre-processing step, *e.g.* abandoning frames that are of too low quality. However, in this situation, the other baseline methods will not be able to be directly trained on the pre-processed data, while our method can still be trained on data unevenly distributed on the timeline. We refer to data unevenly distributed in time as *temporal discontinuities*.

To verify the robustness of our model under temporal discontinuities, we conduct three groups of experiments by randomly masking 0%, 50% and 70% of the frames respectively. Accordingly, we also can easily adapt our loss function to:

$$\min \frac{1}{T}\int_{t=1}^{T} ||\mathbf{v}^t - \hat{\mathbf{v}}^t||_2^2 \approx \frac{1}{T}\sum_{t\in\Phi} ||\mathbf{v}^t - \hat{\mathbf{v}}^t||_2^2 \quad (4)$$

where Φ is the set of the observed frames and $T = |\Phi|$.

The corresponding experimental results are presented in Tab. 1. For both velocity prediction errors (Err_{vel}) and flow prediction errors (Err_{flow}), the performance remains stable across varying masking levels, with only minor variations observed. Notably, even under the 70% masking scenario, the errors are comparable to those without masking. These findings highlight the model's capacity to maintain accuracy and adaptability despite significant temporal discontinuities, making it well-suited for real-world applications where data observations are sparse or unevenly distributed in time.

2.2. Input requirements and parameter efficiency

Our approach demonstrates a significant advantage in both input requirements and parameter efficiency. PredFlow requires four consecutive RGB frames and has 42.534M parameters, HINN requires two consecutive RGB frames and has 8.374M parameters, while PredRNNv2, SimVP, and TAU demand ten consecutive frames, with 24.216M, 18.604M, and 44.657M parameters, respectively. Although Baseline_I operates on a single optical flow frame, it needs laborious hand-tuning. Our model achieves mostly the best long-horizon prediction with only a single optical flow frame as input, and merely 3.06M parameters. Vastly different from other models, our input contains little dynamics. The substantial reduction in parameter count, coupled with the minimal input requirements, highlights our model's efficiency, especially when data and computational resources are limited.

Std	Err_{vel}	Err_{flow}	Prob	Err_{vel}	Err_{flow}	G/U	Err_{vel}	Err_{flow}
0.1	1.0437	1.6281	0.1	1.0544	1.6380	0.3/0.7	1.0709	1.6476
1	1.0615	1.6438	0.7	1.0491	1.6347	0.5/0.5	1.0808	1.6523
10	1.0639	1.6484	1	1.0759	1.6526	0.7/0.3	1.0669	1.6451

Table 2. Evaluation with noises (1-3) on Drill_2. 1: a zero-mean Gaussian noise with a Std (standard deviation); 2: a 2D uniform noise with Prob being the probability of values within [-0.7, 0.7] × [-0.8, 0.8]; 3: a mixture noise as the weighted sum of the Gaussian (G) and the uniform (U) noise.

2.3. Robustness to Input Noise

CCTV camera data is by far the major data on high-density crowd particularly because of the uninvasiveness of the data collection process. However, these videos often suffer from low quality because they are captured from distant cameras and tend to be blurred, making optical flow estimation unreliable. Owning to the physics model in Crowd MPM, our model incorporates a self-correction strategy (P2G operation) that effectively handles spatially noisy.

Since the noise in optical flows is unknown, we conducted an experiment on Drill_2 to show that our method is robust to several noises with various shapes and magnitudes, *e.g.* Gaussian, uniform, and their mixtures, in Tab. 2. All results are similar, demonstrating the robustness of our model. To understand this, when noisy data are given, the P2G projects it onto the grid (*i.e.* equivalent to one step smoothing) then the mass and momentum are preserved the whole time, essentially filtering out high-frequency noises. During training, even when the ground-truth is noisy, Crowd MPM computes its closest physically sensible prediction, by conserving the mass and momentum. Admittedly, if the optical flow is completely obscure,

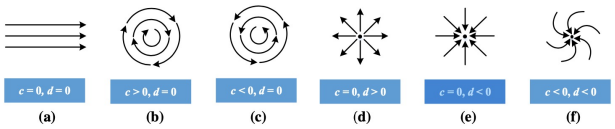

Figure 3. a–e Five simple motion vector fields corresponding to c and d (where c and d denote the value of curl and divergence respectively)[13].(a) no rotation or dispersal/gathering, (b) the rotation is clockwise and there is no divergence, (c) the rotation is counterclockwise without any dispersal/gathering, (d) the motion is pure dispersal without any curl component, (e) the motion is pure gathering without any curl component, (f) the motion is relatively complex since it has rotation and convergence simultaneously.

e.g. a white noise, Crowd MPM would not learn the underlying dynamics. In this case, Crowd MPM can still be hand-tuned to visually mimic the data.

2.4. Visual Analysis Based on Learned Crowd Materials

In addition to prediction and simulation, our method is also a good tool for visual analysis of high-density crowds, especially in terms of its global flows. Previously, operators such as curl (vorticity) and divergence have been proposed as a good visualization tool for analysis [13]. However, [13] proposes to compute vorticity and divergence on optical flows while our method compute them on the velocity fields. Curl is defined as $c = \frac{\partial \mathbf{v}_y}{\partial x} - \frac{\partial \mathbf{v}_x}{\partial y}$, representing the tendency of the flow to exhibit rotational motion or vorticity. Divergence, on the other hand, is defined as $d = \frac{\partial \mathbf{v}_x}{\partial x} + \frac{\partial \mathbf{v}_y}{\partial y}$, representing the rate at which a quantity (such as mass or velocity) expands or contracts at a point. Note that $\mathbf{v}$ is a motion vector field, which can either be optical flow or velocity field. Several examples of global flow patterns are provided in Fig. 3, illustrating how these flows can be analyzed and interpreted using curl and divergence.

In our scenario, curl reflects steering behaviors, showing how individuals adjust their movements, particularly at the periphery or near obstacles or other peoople. Divergence indicate areas where the crowd density is decreasing (positive divergence) or increasing (negative divergence), such as the region near an exit in Drill where people move towards. We show a qualitative comparison on Drill in Fig. 4 at $3s$, $9s$ and $15s$.

[13] is designed for crowd behavior classification, such as lane formation, clockwise arch and so on, so it aims to recognize motion patterns based on normalized velocities (per pixel), shown in iii and iv in Fig. 4 (zoom in for a better view). Therefore, the curl and divergence computed based on normalized velocities can only indicate the general patterns of motions, not the actual motions. Visually, the global flow patterns are not obvious.

In contrast, our model can act as a more detailed analyzer based on the velocity and interpret how the behavior evolves. For example, the areas pointed out by arrows in Fig. 4a vi, Fig. 4b vi and Fig. 4c vi have relatively high positive divergence. Physically, this means there are masses flowing out of these areas. In time, these areas move toward the exit. This captures that the fact that people come into the scene and walk towards the exit. After a short while, there is no people entering the scene. Correspondingly, the high divergence areas first appear near the top entrance then gradually move towards the exit. The divergence in front of the exit is always negative, *i.e*. masses flowing in, due to that for this area the exit rate is lower than the entering rate (*i.e*. people aggregating in front of the exit but not many can get through), hence the blue regions.

In terms of the curl, it reflects the local rotations of masses in a region. This is reflected by the red and blue areas near the exit at $t = 3s$ in Fig. 4a v. This is when the first batch of people arrive at the exit. At this time, people can still go through the exit relatively easily, especially the people right in front of the exit walking relatively faster than the people on both sides of the exit. This caused a relative local rotation of the mass flow, *i.e*. masses in the middle flowing out faster than the ones on both sides, and masses on both sides getting into the middle flow. Both the clockwise (red region) and counter-clockwise (blue region) vorticity reflect this. This curl becomes less prominent at $t = 9s$ and $t = 12$, when many people are crowded in front of the exit and start to block the way, so that the general movements become slow.

Fig. 4 demonstrates the importance of being able to extract and stably simulate the velocity fields for analysis. Capturing the underlying velocity field provides clear global flow trends, then direct analysis on the optical flows which might be polluted by noises [13]. Our method provides a tool which not only extracts and simulates the velocity field, but also learns the dynamics from specific crowds.

2.5. Ablation Study

We analyze the impact of different components, by evaluating four variants of our model, including different combinations of learnable parameters:

- ***Baseline***$_I$, a weakly incompressible fluid model [7], with a global, non-learnable parameter ϵ whose value is obtained by a grid search.
- ***Baseline***$_{II}$ is the same as ***Baseline***$_I$ but with ϵ learnable for each particle.
- ***Baseline***$_{III}$ has a new strain-stress tensor and new learnable parameter k, based on ***Baseline***$_{II}$.
- ***Baseline***$_{IV}$ is ***Baseline***$_{III}$ with NN_α (introduced in Sec.3.4 in main paper).
- ***Ours*** is the full model.

Table 3 clearly shows a continuously improvement across multiple metrics, when more components are added. It

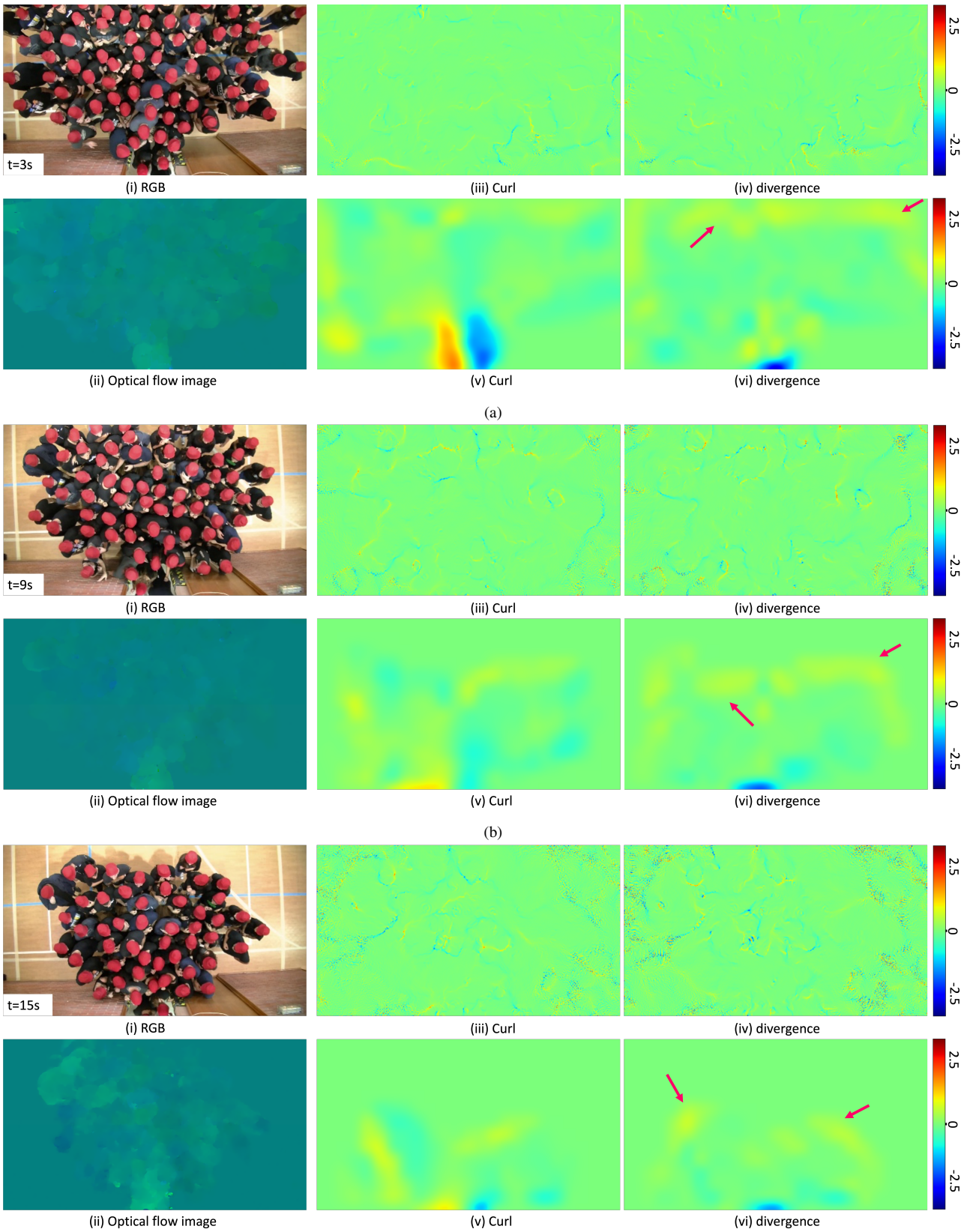

Figure 4. Qualitative comparison on curl and divergence map. iii and iv are generated from [13], while v and vi are obtained by our method. Zoom in for a better view.

Methods	Drill_1		Hajj	
	Err_{vel}	Err_{flow}	Err_{vel}	Err_{flow}
Baseline$_I$	0.60740	0.98405	1.45878	1.68066
Baseline$_{II}$	0.60727	0.98396	1.45709	1.67930
Baseline$_{III}$	0.60483	0.98160	1.45679	1.67906
Baseline$_{IV}$	0.48987	0.88263	0.76161	1.07112
Ours(mean)	0.48276	0.87165	0.69135	0.99511
Ours(best)	0.48161	0.87033	0.68281	0.98604

Table 3. Ablation study on $Drill_1$ and $Hajj$. Results involving active force are based on 10 trials. The training and testing duration are 60 frames on $Drill_1$ and 30 frames on $Hajj$.

has been argued that high-density crowds behave like fluids [10]. This is confirmed by the hand-tuned ***Baseline***$_I$, whose performance is reasonable. ***Baseline***$_{II}$ shows the necessity of making the material learnable. With learnable ϵ, the material becomes more heterogeneous and fits the data better. Furthermore, an addition of a learnable compression component (***Baseline***$_{III}$) can capture the dynamics better. We notice that the difference between ***Baseline***$_{I-III}$ seems small. After further investigation, this is likely to be caused by two factors. First, ***Baseline***$_I$ is a already good baseline as argued by previous research. Hajj crowds form stable flows circling around the center of the scene. Therefore, it is possible to hand-tune ***Baseline***$_I$ to mimic the general dynamics, although this relies on heavy human labor. Second, the metrics are based on the optical flow or the velocity field estimated from optical flow, where the scale of values is small.

The results are significantly after active forces are introduced, even with only the motion alignment component in ***Baseline***$_{IV}$ with no stochasticity. This further proves such crowds can be interpreted as active matters and learning such active forces greatly enhance the model's ability to predict dynamics. Further adding the stochastic part again significantly improves the results (***Ours***).

3. Crowd Material Point Method

There are two mainstream methods to solve PDEs. One is the Lagrangian method which treats the material as a set of particles and tracks them to determine their properties such as mass, positions, and velocities. The other one is the Eulerian method which discretizes the space into grids containing nodes with various properties and employs the grids to represent materials. Material Point Method (MPM) is a hybrid Eulerian-Lagrangian method combining the advantages of Eulerian and Lagrangian methods. As our model involves Eulerian data and Lagrangian behaviors, we enhance the standard MPM to propose Crowd MPM to simulate extremely high-density crowds.

Our derivations are mainly following [4]. The main changes are brought by our newly introduced learnable stress and active forces. Similar to the standard MPM, our Crowd MPM also incorporates both the Lagrangian view and the Eulerian view. Under the Lagrangian view, we focus on the particle p with mass m_p, position $\mathbf{x}_p$, and velocity $\mathbf{v}_p$. The motion of material is defined by a deformation map $\Phi(\cdot, t) : \Omega^0 \to \Omega^t$, where $\Omega^0, \Omega^t \subset \mathbb{R}^2$. Ω^t denotes the set of positions of particles at time t. As a result, Ω^0 is the set of initial positions of particles. Particularly, we use $\mathbf{X}_p$ and $\mathbf{x}_p$ to denote the initial position and the current position for any particle in the simulated material, respectively. We can note that $\mathbf{X}_p = \mathbf{x}_p^0$. The deformation map Φ further enables us to determine the velocity (we ignore the subscript p for notation simplicity in this section):

$$\mathbf{V}(\mathbf{X}, t) = \frac{\partial \Phi}{\partial t}(\mathbf{X}, t). \tag{5}$$

The Jacobian of the deformation map Φ is significantly important and will be used often later:

$$\mathbf{F}(\mathbf{X}, t) = \frac{\partial \Phi}{\partial \mathbf{X}}(\mathbf{X}, t). \tag{6}$$

The most common way to employ $\mathbf{F}$ is to use its determinant J. Finally, we give the governing equations based on the conservation of mass and conservation of momentum:

$$R(\mathbf{X}, t)J(\mathbf{X}, t) = R(\mathbf{X}, 0), \tag{7}$$

$$R(\mathbf{X}, 0)\frac{\partial \mathbf{V}}{\partial t} = \nabla^{\mathbf{X}} \cdot \mathbf{P}^{\mathbf{cm}} + R(\mathbf{X}, 0)(\mathbf{B}^{\mathbf{bd}} + \mathbf{B}^{\mathbf{act}}), \tag{8}$$

where R is the Lagrangian mass density, $\mathbf{P}^{\mathbf{cm}}$ is the crowd material stress under the Lagrangian view, and $\mathbf{B}^{\mathbf{bd}}$ and $\mathbf{B}^{\mathbf{act}}$ denote the accelerations deriving from the body force and the active force on $\mathbf{X}$, respectively. To be more specific, the Lagrangian mass density is defined via:

$$R(\mathbf{X}, t) = \rho(\Phi(\mathbf{X}, t), t) = \rho(\mathbf{x}, t), \tag{9}$$

$$\rho(\mathbf{x}, t) = \lim_{\epsilon \to +0} \frac{mass(B_\epsilon^t)}{\int_{B_\epsilon^t} d\mathbf{x}}, \tag{10}$$

where $B_\epsilon^t \subset \Omega^t$ is a ball with radius ϵ and center $x \in \Omega^t$. We note that the Lagrangian view builds governing equations on material space Ω^0. We offer the proof for Eq. (7) and Eq. (8) in Sec. 4.

Under the Eulerian view, we pay attention to the world space Ω^t. In addition, grids consisting of a series of nodes with index $\mathbf{i}$ are built to estimate material motion. Each note $\mathbf{i}$ has physics properties including mass $m_{\mathbf{i}}$, position $\mathbf{x}_{\mathbf{i}}$, and velocity $\mathbf{v}_{\mathbf{i}}$, *etc*. We also give the governing equations based on the conservation of mass and conservation of momentum:

$$\frac{D}{Dt}\rho(\mathbf{x}, t) + \rho(\mathbf{x}, t)\nabla^{\mathbf{x}} \cdot \mathbf{v}(\mathbf{x}, t) = 0 \tag{11}$$

$$\rho(\mathbf{x},t)\frac{D\mathbf{v}}{Dt} = \nabla^{\mathbf{x}}\cdot\sigma^{\mathbf{cm}} + \rho(\mathbf{x},t)(\mathbf{b}^{\mathbf{bd}}+\mathbf{b}^{\mathbf{act}}) \quad (12)$$

where $\frac{D}{Dt}$ is the material derivative [9], $\mathbf{v}(\mathbf{x},t) = \mathbf{V}(\Phi^{-1}(\mathbf{x},t),t)$, $\sigma^{\mathbf{cm}}$ is the crowd material stress under the Eulerian view, and $\mathbf{b}^{\mathbf{bd}}$ and $\mathbf{b}^{\mathbf{act}}$ denote the accelerations deriving from the body force and the active force on $\mathbf{x}$, respectively. We also leave the proof for Eq. (11) and Eq. (12) in Sec. 4

Finally, we give the weak form, which is crucial for the discretization, of the conservation of momentum in the Eulerian view:

$$\begin{aligned}\int_{\Omega^t} q_i\rho a_i d\mathbf{x} = &\int_{\partial\Omega^t} q_i t_i ds(\mathbf{x}) - \int_{\Omega^t} q_{i,k}\sigma^{cm}_{ik} d\mathbf{x} \\ &+ \int_{\Omega^t} q_i\rho b^{bd}_i d\mathbf{x} + \int_{\Omega^t} q_i\rho b^{act}_i d\mathbf{x}. \quad (13)\end{aligned}$$

where q_i is the ith component of an arbitrary function $\mathbf{q}(\cdot,t):\Omega^t\to\mathbb{R}^d$, $q_{i,k}=\frac{\partial q_i}{\partial x_k}$, a_i is the ith component of $\boldsymbol{a}=\frac{D\mathbf{v}}{Dt}$, and t_i is the ith component of the boundary force per unit reference area $\mathbf{t}(\mathbf{x},t)$. Similarly, σ^{cm}_{ik}, b^{bd}_i and b^{act}_i are the components of $\sigma^{\mathbf{cm}}$, $\mathbf{b}^{\mathbf{bd}}$, and $\mathbf{b}^{\mathbf{act}}$, respectively. ds denotes a tiny area. For efficient expression, the summation is implied on the repeated index. The proof of the weak form can be found in Sec. 4.

Subsequently, we can clarify three steps in our Crowd MPM individually: (1) Particle-to-grid transfer in Sec. 3.1 (2) Grid Operation in Sec. 3.2, and (3) Grid-to-particle transfer in Sec. 3.3.

3.1. Particle-to-grid Transfer

This step aims to transfer particle mass and momentum to the grid at each timestamp n. Each particle contributes to the transfer process based on weights relying on particles and grids. Specifically, the transfer process in defined by:

$$m^n_{\mathbf{i}} = \int_{\mathbf{x}\in\Omega^n} W^n_{\mathbf{i}}(\mathbf{x})\rho(\mathbf{x})\,d\mathbf{x}, \quad (14)$$

$$m^n_{\mathbf{i}}\mathbf{v}^n_{\mathbf{i}} = \int_{\mathbf{x}\in\Omega^n} W^{n'}_{\mathbf{i}}(\mathbf{x})\rho(\mathbf{x})\mathbf{v}(\mathbf{x})d\mathbf{x}, \quad (15)$$

where $W^n_{\mathbf{i}}$ and $W^{n'}_{\mathbf{i}}$ are weight functions, Ω^n is the set of all particles at timestamp n, ρ denotes density.

We follow the Affine Particle-In-Cell (APIC) method [3] to discretize Eq. (14) and Eq. (15) due to its excellent numerical properties:

$$m^n_{\mathbf{i}} = \sum_p w^n_{\mathbf{i}p} m_p, \quad (16)$$

$$m^n_{\mathbf{i}}\mathbf{v}^n_{\mathbf{i}} = \sum_p w^n_{\mathbf{i}p}[m_p\mathbf{v}^n_p + m_p\mathbf{C}^n_p(\mathbf{x}_{\mathbf{i}} - \mathbf{x}^n_p)], \quad (17)$$

where $w^n_{\mathbf{i}p} = \phi(\mathbf{x}_{\mathbf{i}} - \mathbf{x}^n_p)$ (ϕ is a quadratic B-spline function [2]), $\mathbf{C}^n_p$ is the affine velocity gradient [3]. m_p and $\mathbf{x}_{\mathbf{i}}$ don't have superscript n because m_p doesn't change with time and the grids are rebuilt at each MPM iteration.

3.2. Grid Operation

We update velocities on the grid by:

$$\mathbf{v}^{n+1}_{\mathbf{i}} = \mathbf{v}^n_{\mathbf{i}} + \Delta t\frac{\mathbf{f}^n_{\mathbf{i}}}{m_{\mathbf{i}}}, \quad (18)$$

Then boundary conditions are employed to refine the updated velocities:

$$\mathbf{v}^{n+1}_{\mathbf{i}} = \mathbf{BC}(\mathbf{v}^{n+1}_{\mathbf{i}}) = \mathbf{v}^{n+1}_{\mathbf{i}} - \gamma\mathbf{n}\left\langle\mathbf{n},\mathbf{v}^{n+1}_{\mathbf{i}}\right\rangle. \quad (19)$$

We explain how to obtain $\mathbf{v}^n_{\mathbf{i}}$ and $m_{\mathbf{i}}$ in the last section. Next, I will introduce how to estimate $\mathbf{f}^n_{\mathbf{i}}$.

Like Sec. 3.1, we transfer forces on particles to the grid at each timestamp n. We give the transfer process for each component of forces. To avoid confusion, we use Greek letters like α and β to denote the component index.

$$\mathbf{f}^n_{\mathbf{i}\alpha} = \int_{\mathbf{x}\in\Omega^n} W^n_{\mathbf{i}}(\mathbf{x})\rho(\mathbf{x},t)\boldsymbol{a}_\alpha(\mathbf{x},t)d\mathbf{x} \quad (20)$$

where other components of $\mathbf{f}^n_{\mathbf{i}}$ has a similar transfer process. According to the weak form of the conservation of momentum in the Eulerian view Eq. (13), we have:

$$\begin{aligned}\mathbf{f}_{\mathbf{i}\alpha} = \int_{\Omega^t} W_{\mathbf{i}}\rho a_\alpha d\mathbf{x} = &\int_{\partial\Omega^t} W_{\mathbf{i}} t_\alpha ds(\mathbf{x}) - \int_{\Omega^t} W_{\mathbf{i},\gamma}\sigma^{cm}_{\alpha\gamma} d\mathbf{x} \\ &+ \int_{\Omega^t} W_{\mathbf{i}}\rho b^{bd}_\alpha d\mathbf{x} + \int_{\Omega^t} W_{\mathbf{i}}\rho b^{act}_\alpha d\mathbf{x}.\end{aligned} \quad (21)$$

where the superscript is ignored for notation simplicity. Here, we set $q_\alpha = W_{\mathbf{i}}$ and $q_\gamma = 0$ because $\mathbf{q}$ is arbitrary. Following previous work [4], we omit the $\int_{\partial\Omega^t} W_{\mathbf{i}} t_\alpha ds(\mathbf{x})$ in Eq. (21) for simplicity given its minor effect. Given our estimation of $\sigma^{\mathbf{cm}}_{\mathbf{p}} = \sigma(\mathbf{x}_p,t)$ at every Lagrangian particle $\mathbf{x}_p$, we have:

$$\begin{aligned}f^{st}_{\mathbf{i}\alpha} = -\int_{\Omega^t} W_{\mathbf{i},\gamma}\sigma^{cm}_{\alpha\gamma} d\mathbf{x} &\approx -\sum_p \sigma^{cm}_{p\alpha\gamma} w_{\mathbf{i}p,\gamma} V_p \\ &= -\sum_p \sigma^{\mathbf{cm}}_{\mathbf{p}\alpha}\nabla w_{\mathbf{i}p} V_p, \quad (22)\end{aligned}$$

where $\sigma^{\mathbf{cm}}_{\mathbf{p}\alpha}$ is the αth row of $\sigma^{\mathbf{cm}}_{\mathbf{p}}$, V_p is the volume of $B^t_{\Delta x}$ at the Lagrangian particle $\mathbf{x}_p$. We follow [3] to estimate V_p. Because we use the quadratic B-spline function to represent $w_{\mathbf{i}p}$, we can obtain its gradient by:

$$\nabla w_{\mathbf{i}p} = \frac{4}{\Delta x^2} w_{\mathbf{i}p}(\mathbf{x}_{\mathbf{i}} - \mathbf{x}_p) \quad (23)$$

Combining Eq. (22) and Eq. (23) results in:

$$f^{st}_{\mathbf{i}\alpha} \approx -\sum_p \frac{4}{\Delta x^2} w_{\mathbf{i}p}\sigma^{\mathbf{cm}}_{\mathbf{p}\alpha}(\mathbf{x}_{\mathbf{i}} - \mathbf{x}_p)V_p \quad (24)$$

By defining:

$$\frac{4}{\Delta x^2}\sigma^{\mathbf{cm}}_{\mathbf{p}\alpha}(\mathbf{x_i}-\mathbf{x}_p)V_p = \mathbf{G}_p(\mathbf{x}_i-\mathbf{x}_p)+\sum_{p'\in N_p}(\boldsymbol{f}_r^{pp'}-\boldsymbol{f}_t^{pp'}), \tag{25}$$

we can get the Eq.11 in the main paper. For the body force, we take the centripetal force as an example. Other body forces such as the goal attraction force have a similar derivation. Therefore, $\mathbf{b}^{\mathbf{bd}}(\mathbf{x},t)=\frac{\mathbf{v}(\mathbf{x},t)^2}{r}$, where r is the length of the radius of the moving circle. Then, we have:

$$f^{bd}_{\mathbf{i}\alpha}=\int_{\Omega^t}W_{\mathbf{i}}\rho b^{bd}_\alpha d\mathbf{x}=\int_{\Omega^t}W_{\mathbf{i}}\rho\frac{\mathbf{v}^2_\alpha}{r}d\mathbf{x} \approx\sum_p w_{\mathbf{i}p}m_p\frac{\mathbf{v}^2_{p\alpha}}{r}. \tag{26}$$

As for the active force, we have $\mathbf{b}^{\mathbf{act}}=a\mathbf{v}+\mathbf{c}$, where $\mathbf{c}$ is the stochastic part of the active force, *i.e.* $\mathbf{c}=-b|\mathbf{v}|^2\mathbf{v}+D_L\nabla(\nabla\cdot\mathbf{v})+D_1\nabla^2\mathbf{v}+D_2(\mathbf{v}\cdot\nabla)^2\mathbf{v}+\tilde{\mathbf{f}}$. a and $\mathbf{c}$ are predicted by neural networks. Then, we have:

$$f^{act}_{\mathbf{i}\alpha}=\int_{\Omega^t}W_{\mathbf{i}}\rho b^{act}_\alpha d\mathbf{x}=\int_{\Omega^t}W_{\mathbf{i}}\rho(a\mathbf{v}_\alpha+c_\alpha)d\mathbf{x} =\sum_p w_{\mathbf{i}p}m_p(a\mathbf{v}_{p\alpha}+c_{p\alpha}). \tag{27}$$

Combining Eq. (24), Eq. (26) and Eq. (27), we have:

$$\mathbf{f}_{\mathbf{i}\alpha}=f^{st}_{\mathbf{i}\alpha}+f^{bd}_{\mathbf{i}\alpha}+f^{act}_{\mathbf{i}\alpha}\approx\sum_p-\frac{4}{\Delta x^2}w_{\mathbf{i}p}\sigma^{\mathbf{cm}}_{\mathbf{p}\alpha}(\mathbf{x_i}-\mathbf{x}_p)V_p +w_{\mathbf{i}p}m_p\frac{\mathbf{v}^2_{p\alpha}}{r}+w_{\mathbf{i}p}m_p(a\mathbf{v}_{p\alpha}+c_{p\alpha}) \tag{28}$$

3.3. Grid-to particle Transfer

Finally, we transfer the updated velocities $\mathbf{v}_\mathbf{i}^{n+1}$ back to particles and update positions of these particles:

$$\mathbf{v}_p^{n+1}=\sum_{\mathbf{i}}w^n_{\mathbf{i}p}\mathbf{v}^{n+1}_{\mathbf{i}},\ \mathbf{x}_p^{n+1}=\mathbf{x}_p^n+\Delta t\mathbf{v}_p^{n+1}, \tag{29}$$

We update the velocity gradient $\mathbf{C}_p^{n+1}$ for next step:

$$\mathbf{C}_p^{n+1}=\frac{4}{\Delta x^2}\sum_i w^n_{\mathbf{i}p}\mathbf{v}^{n+1}_{\mathbf{i}}(\mathbf{x_i}-\mathbf{x}_p^n)^T. \tag{30}$$

Additionally, we also update the deformation gradient $\mathbf{F}_p^{n+1}$ for estimating V_p:

$$\mathbf{F}_p^{n+1}=(\mathbf{I}+\Delta t\mathbf{C}_p^{n+1})\mathbf{F}_p^n. \tag{31}$$

4. Proof Details

Push Forward and Pull Back. We introduce the push forward and pull back operations based on the deformation map Φ for efficient formula derivation and understanding. The deformation map $\Phi:\Omega^0\to\Omega^t$ typically is assumed to be bijective, which means that any function defined on one set like Ω^0 can naturally be regarded as another function defined on another set like Ω^t by changing variables. Push forward and pull back are two ways of variable substitution here. To be formal, given the time t and a function $G(\cdot,t):\Omega^0\to\mathbb{R}$, we define the push forward $g(\cdot,t):\Omega^t\to\mathbb{R}$ as $g(\mathbf{x},t)=G(\Phi^{-1}(\mathbf{x},t),t)$ with $\mathbf{x}\in\Omega^t$. Similarly, given the time t and a function $g(\cdot,t):\Omega^t\to\mathbb{R}$, the pull back $G(\cdot,t):\Omega^0\to\mathbb{R}$ is defined as $G(\mathbf{X},t)=g(\Phi(\mathbf{X},t),t)$ with $\mathbf{X}\in\Omega^0$. Particularly, the pull back of the pull forward of a function such as $G(\cdot,t)$ is itself. We commonly think of a function of $\mathbf{X}$/$\mathbf{x}$ as Lagrangian/Eulerian. Therefore, the push forward (pull back) of a function which is Lagrangian (Eulerian) is Eulerian (Lagrangian).

Conservation of Mass under the Lagrangian View. For an arbitrary $\mathbf{x}\in\Omega^t$, we consider the ball $B^t_\epsilon\subset\Omega^t$ with radius ϵ and center $\mathbf{x}$. The mass in B^t_ϵ shouldn't vary over time when ϵ is small enough. Therefore, we have $mass(B^t_\epsilon)=mass(B^0_\epsilon)$. According to the definition of ρ, R, and J, we have:

$$mass(B^t_\epsilon)=\int_{B^t_\epsilon}\rho(\mathbf{x},t)d\mathbf{x}=\int_{B^0_\epsilon}R(\mathbf{X},t)Jd\mathbf{X}, \tag{32}$$

$$mass(B^0_\epsilon)=\int_{B^0_\epsilon}R(\mathbf{X},0)d\mathbf{X}, \tag{33}$$

for all $B^t_\epsilon\subset\Omega^t$ $(t\geq 0)$, where the second equality in Eq. (32) is obtained by the formula of changing variables. From Eq. (32) and Eq. (33), we can derive:

$$\int_{B^0_\epsilon}R(\mathbf{X},t)Jd\mathbf{X}=\int_{B^0_\epsilon}R(\mathbf{X},0)d\mathbf{X}. \tag{34}$$

Given that B^t_ϵ is arbitrary, we can obtain:

$$R(\mathbf{X},t)J(\mathbf{X},t)=R(\mathbf{X},0), \tag{35}$$

for any $\mathbf{X}\in\Omega^0$.

Conservation of Mass under the Eulerian View. We start from Eq. (35), *i.e.* Eq. (7), to derive the conservation of mass equation under the Eulerian view. First, we have:

$$\frac{\partial}{\partial t}(R(\mathbf{X},t)J(\mathbf{X},t))=\frac{\partial}{\partial t}R(\mathbf{X},0)=0, \tag{36}$$

according to Eq. (35). Then, it is easy to get:

$$\frac{\partial}{\partial t}(RJ)=\frac{\partial R}{\partial t}J+R\frac{\partial J}{\partial t}=0, \tag{37}$$

where we neglect $(\mathbf{X}, t)$ for notation simplicity. Then, we use the result from [4]:

$$\frac{\partial J}{\partial t} = J\frac{\partial v_1}{\partial x_1} + J\frac{\partial v_2}{\partial x_2}, \tag{38}$$

where $\mathbf{x} = (x_1, x_2) \in \Omega^t$ and $\mathbf{v}(\mathbf{x}, t) = (v_1, v_2)$. Combining Eq. (37) and Eq. (38) results in:

$$\frac{\partial R}{\partial t}J + RJ(\frac{\partial v_1}{\partial x_1} + \frac{\partial v_2}{\partial x_2}) = 0. \tag{39}$$

We push forward on both sides of Eq. (39) to obtain:

$$\frac{D}{Dt}\rho(\mathbf{x}, t) + \rho(\mathbf{x}, t)\nabla^{\mathbf{x}} \cdot \mathbf{v}(\mathbf{x}, t) = 0. \tag{40}$$

Conservation of Momentum under the Eulerian View. Given an arbitrary $B_\epsilon^t \subset \Omega^t$, the momentum change on B_ϵ^t can be expressed as:

$$\frac{d}{dt}\int_{B_\epsilon^t} \rho(\mathbf{x}, t)\mathbf{v}(\mathbf{x}, t)d\mathbf{x} = \int_{\partial B_\epsilon^t} \sigma^{\mathbf{cm}}\mathbf{n}ds(\mathbf{x}) + \int_{B_\epsilon^t} \rho(\mathbf{x}, t)(\mathbf{b}^{\mathbf{bd}} + \mathbf{b}^{\mathbf{act}})d\mathbf{x}, \tag{41}$$

where $\mathbf{n}(\mathbf{x})$ is the unit outward normal of ∂B_ϵ^t at $\mathbf{x}$ and ds denotes a tiny area. We further have:

$$\frac{d}{dt}\int_{B_\epsilon^t} \rho(\mathbf{x}, t)\mathbf{v}(\mathbf{x}, t)d\mathbf{x} = \frac{d}{dt}\int_{B_\epsilon^0} R(\mathbf{X}, t)\mathbf{V}(\mathbf{X}, t)Jd\mathbf{X} = \int_{B_\epsilon^0} R(\mathbf{X}, t)J(\mathbf{X}, t)\mathbf{A}(\mathbf{X}, t)d\mathbf{X}, \tag{42}$$

where $\mathbf{A}(\mathbf{X}, t) = \frac{\partial^2 \Phi}{\partial t^2}(\mathbf{X}, t) = \frac{\partial \mathbf{V}}{\partial t}(\mathbf{X}, t)$, the first equality comes from the formula of changing variables, and the second equality obtained by swapping the order of the taking derivative symbol and the integral symbol. Combining Eq. (41) and Eq. (42) results in:

$$\int_{B_\epsilon^0} R(\mathbf{X}, t)J(\mathbf{X}, t)\mathbf{A}(\mathbf{X}, t)d\mathbf{X} = \int_{\partial B_\epsilon^t} \sigma^{\mathbf{cm}}\mathbf{n}ds(\mathbf{x}) + \int_{B_\epsilon^t} \rho(\mathbf{x}, t)(\mathbf{b}^{\mathbf{bd}} + \mathbf{b}^{\mathbf{act}})d\mathbf{x} \tag{43}$$

We push forward the left side of Eq. (43) to get:

$$\int_{B_\epsilon^0} R(\mathbf{X}, t)J(\mathbf{X}, t)\mathbf{A}(\mathbf{X}, t)d\mathbf{X} = \int_{B_\epsilon^t} \rho(\mathbf{x}, t)\mathbf{a}(\mathbf{x}, t)d\mathbf{x}, \tag{44}$$

where $\mathbf{a}(\mathbf{x}, t) = \mathbf{A}(\Phi^{-1}(\mathbf{x}, t), t)$. Further, we also have $\mathbf{A}(\Phi^{-1}(\mathbf{x}, t), t) = \frac{D\mathbf{v}}{Dt}$, which has proof in [4]. Then, Eq. (43) becomes:

$$\int_{B_\epsilon^t} \rho(\mathbf{x}, t)\mathbf{a}(\mathbf{x}, t)d\mathbf{x} = \int_{B_\epsilon^t} \rho(\mathbf{x}, t)\frac{D\mathbf{v}}{Dt}d\mathbf{x} = \int_{\partial B_\epsilon^t} \sigma^{\mathbf{cm}}\mathbf{n}ds(\mathbf{x}) + \int_{B_\epsilon^t} \rho(\mathbf{x}, t)(\mathbf{b}^{\mathbf{bd}} + \mathbf{b}^{\mathbf{act}})d\mathbf{x} = \int_{B_\epsilon^t} \nabla^{\mathbf{x}} \cdot \sigma^{\mathbf{cm}}d\mathbf{x} + \int_{B_\epsilon^t} \rho(\mathbf{x}, t)(\mathbf{b}^{\mathbf{bd}} + \mathbf{b}^{\mathbf{act}})d\mathbf{x} \tag{45}$$

Since B_ϵ^t is arbitrary, we have:

$$\rho(\mathbf{x}, t)\frac{D\mathbf{v}}{Dt} = \nabla^{\mathbf{x}} \cdot \sigma^{\mathbf{cm}} + \rho(\mathbf{x}, t)(\mathbf{b}^{\mathbf{bd}} + \mathbf{b}^{\mathbf{act}}). \tag{46}$$

Conservation of Momentum under the Lagrangian View. We can also derive the conservation of momentum under the Lagrangian view from Eq. (43). Instead of pushing forward, we pull back the right side of Eq. (43). The pull back of the first integral on the right side of Eq. (43) is:

$$\int_{\partial B_\epsilon^t} \sigma^{\mathbf{cm}}(\mathbf{x}, t)\mathbf{n}ds(\mathbf{x}) = \int_{\partial B_\epsilon^0} J(\mathbf{X}, t)\sigma^{\mathbf{cm}}(\Phi(\mathbf{X}, t), t)\mathbf{F}^{-T}(\mathbf{X}, t)\mathbf{N}(\mathbf{X})ds(\mathbf{X}), \tag{47}$$

where $\mathbf{N}(\mathbf{X})$ is the unit outward normal of ∂B_ϵ^0 at $\mathbf{X}$. We introduce our the Crowd first Piola Kirchoff stress $\mathbf{P}^{\mathbf{cm}} = J\sigma^{\mathbf{cm}}\mathbf{F}^{-T}$ and get:

$$\int_{\partial B_\epsilon^t} \sigma^{\mathbf{cm}}(\mathbf{x}, t)\mathbf{n}ds(\mathbf{x}) = \int_{\partial B_\epsilon^0} \mathbf{P}^{\mathbf{cm}}(\mathbf{X}, t)\mathbf{N}ds(\mathbf{X}) = \int_{B_\epsilon^0} \nabla^{\mathbf{x}} \cdot \mathbf{P}^{\mathbf{cm}}(\mathbf{X}, t)d\mathbf{X}. \tag{48}$$

Subsequently, we pull back the second integral on the right side of Eq. (43):

$$\int_{B_\epsilon^t} \rho(\mathbf{x}, t)(\mathbf{b}^{\mathbf{bd}} + \mathbf{b}^{\mathbf{act}})d\mathbf{x} = \int_{B_\epsilon^0} R(\mathbf{X}, t)J(\mathbf{X}, t)(\mathbf{B}^{\mathbf{bd}} + \mathbf{B}^{\mathbf{act}})d\mathbf{X} = \int_{B_\epsilon^0} R(\mathbf{X}, 0)(\mathbf{B}^{\mathbf{bd}} + \mathbf{B}^{\mathbf{act}})d\mathbf{X}, \tag{49}$$

where we use Eq. (35) to get the second equality. We note that the left side of Eq. (43) can be written as using Eq. (35):

$$\int_{B_\epsilon^0} R(\mathbf{X}, t)J(\mathbf{X}, t)\mathbf{A}(\mathbf{X}, t)d\mathbf{X} = \int_{B_\epsilon^0} R(\mathbf{X}, 0)\mathbf{A}(\mathbf{X}, t)d\mathbf{X} = \int_{B_\epsilon^0} R(\mathbf{X}, 0)\frac{\partial \mathbf{V}}{\partial t}(\mathbf{X}, t)d\mathbf{X}. \tag{50}$$

Combining Eq. (43), Eq. (48), Eq. (49), and Eq. (50) results in:

$$\int_{B_\epsilon^0} R(\mathbf{X},0)\frac{\partial \mathbf{V}}{\partial t}(\mathbf{X},t)d\mathbf{X} = \int_{B_\epsilon^0} \nabla^{\mathbf{x}}\cdot\mathbf{P^{cm}}(\mathbf{X},t)d\mathbf{X} + \int_{B_\epsilon^0} R(\mathbf{X},0)(\mathbf{B^{bd}}+\mathbf{B^{act}})d\mathbf{X}. \tag{51}$$

Because B_ϵ^0 is arbitrary, we can derive:

$$R(\mathbf{X},0)\frac{\partial \mathbf{V}}{\partial t} = \nabla^{\mathbf{X}}\cdot\mathbf{P^{cm}} + R(\mathbf{X},0)(\mathbf{B^{bd}}+\mathbf{B^{act}}). \tag{52}$$

Weak Form of the Conservation of Momentum in Both Views. The weak form of an equation means that we can derive its weak form from the equation but can't derive the equation from its weak form. The weak forms of the conservation of momentum in both views are extremely important to derive our discretized Crowd MPM. We start with the Lagrangian view. We ignore the body force and the active force for simplicity:

$$R(\mathbf{X},0)\mathbf{A}(\mathbf{X},t) = \nabla^{\mathbf{X}}\cdot\mathbf{P^{cm}}. \tag{53}$$

Then, we have:

$$R_0 A_i = \sum_j \frac{\partial P_{ij}^{cm}}{\partial X_j} = \sum_j P_{ij,j}^{cm}, \tag{54}$$

where $R_0 = R(\mathbf{X},0)$, A_i is the ith component of $\mathbf{A}(\mathbf{X},t)$, P_{ij}^{cm} is the element at the ith row and jth column in $\mathbf{P^{cm}}$. For efficient expression, the summation is implied on the repeated index. Therefore, we have:

$$R_0 A_i = P_{ij,j}^{cm}. \tag{55}$$

Then, we derive the weak form of Eq. (53) by computing the dot product between an arbitrary function $\mathbf{Q}(\cdot,t) : \Omega^0 \rightarrow \mathbb{R}^d$ and two sides of Eq. (53), respectively, and integrate over Ω^0:

$$\int_{\Omega^0} Q_i(\mathbf{X},t)R(\mathbf{X},0)A_i(\mathbf{X},t)d\mathbf{X} = \int_{\Omega^0} Q_i(\mathbf{X},t)P_{ij,j}^{cm}(\mathbf{X},t)dX = \int_{\Omega^0} ((Q_i(\mathbf{X},t)P_{ij}^{cm}(\mathbf{X},t))_{,j} - Q_{i,j}(\mathbf{X},t)P_{ij}^{cm}(\mathbf{X},t))d\mathbf{X} = \int_{\partial\Omega^0} Q_i P_{ij}^{cm} N_j ds(\mathbf{X}) - \int_{\Omega^0} Q_{i,j}P_{ij}^{cm}d\mathbf{X}. \tag{56}$$

The $P_{ij}^{cm}N_j$ would be determined by a boundary condition. Assuming that $\mathbf{T}(\mathbf{X},t)$ is the boundary force per unit reference area, we have $T_i = P_{ij}^{cm}N_j$. As a result, we have that for an arbitrary $\mathbf{Q}(\cdot,t) : \Omega^0 \rightarrow \mathbb{R}^d$:

$$\int_{\Omega^0} Q_i R_0 A_i d\mathbf{X} = \int_{\partial\Omega^0} Q_i T_i ds(\mathbf{X}) - \int_{\Omega^0} Q_{i,j}P_{ij}^{cm}dX. \tag{57}$$

Considering the full Eq. (52), *i.e.* Eq. (8), we have the final weak form of the conservation of momentum in the Lagrangian view:

$$\int_{\Omega^0} Q_i R_0 A_i d\mathbf{X} = \int_{\partial\Omega^0} Q_i T_i ds(\mathbf{X}) - \int_{\Omega^0} Q_{i,j}P_{ij}^{cm}d\mathbf{X} + \int_{\Omega^0} Q_i R_0 B_i^{bd} d\mathbf{X} + \int_{\Omega^0} Q_i R_0 B_i^{act} d\mathbf{X}. \tag{58}$$

Then we push forward Eq. (58) to get the weak form of the conservation of momentum in the Eulerian view. The push forward of the left side of Eq. (58) is derived by introducing $\mathbf{q}$ that is the push forward of $\mathbf{Q}$:

$$\int_{\Omega^0} Q_i R_0 A_i d\mathbf{X} = \int_{\Omega^t} q_i(\mathbf{x},t)\rho(\mathbf{x},t)a_i(\mathbf{x},t)d\mathbf{x}. \tag{59}$$

Similarly, let $\mathbf{t}$ be the push forward of $\mathbf{T}$, we have:

$$\int_{\partial\Omega^0} Q_i T_i ds(\mathbf{X}) = \int_{\partial\Omega^t} q_i(\mathbf{x},t)t_i(\mathbf{x},t)ds(\mathbf{x}). \tag{60}$$

Before giving the push forward of the second integral on the right side of Eq. (58), we analyze $Q_{i,j}$:

$$Q_{i,j} = \frac{\partial Q_i}{\partial X_j} = \frac{\partial q_i}{\partial x_k}\frac{\partial x_k}{\partial X_j} = q_{i,k}F_{kj}, \tag{61}$$

where the summation is implied on the repeated index k, F_{kj} is the element in $\mathbf{F}$ defined by Eq. (6), and $\mathbf{x} = \Phi(\mathbf{X},t)$. Recall that $\mathbf{P^{cm}} = J\sigma^{\mathbf{cm}}\mathbf{F}^{-T}$, then we have $\sigma^{\mathbf{cm}} = \frac{1}{J}\mathbf{P^{cm}}\mathbf{F}^T$. Therefore, we the relationship between $\sigma^{\mathbf{cm}}$ and $\mathbf{P^{cm}}$:

$$\sigma_{ik}^{cm} == \frac{1}{J}P_{ij}^{cm}F_{kj}. \tag{62}$$

Combining Eq. (61) and Eq. (62) let us derive the push forward of the second integral on the right side of Eq. (58):

$$\int_{\Omega^0} Q_{i,j}P_{ij}^{cm}d\mathbf{X} = \int_{\Omega^t} q_{i,k}(\mathbf{x},t)\sigma_{ik}^{cm}(\mathbf{x},t)d\mathbf{x}. \tag{63}$$

As $\mathbf{b}^{bd}$ and $\mathbf{b}^{act}$ are the natural push forward of $\mathbf{B}^{bd}$ and $\mathbf{B}^{act}$, it is easy to derive the push forward of the remaining integrals in Eq. (58):

$$\int_{\Omega^0} Q_i R_0 B_i^{bd} d\mathbf{X} = \int_{\Omega^t} q_i(\mathbf{x},t)\rho(\mathbf{x},t)b_i^{bd}(\mathbf{x},t)d\mathbf{x}$$
$$\int_{\Omega^0} Q_i R_0 B_i^{act} d\mathbf{X} = \int_{\Omega^t} q_i(\mathbf{x},t)\rho(\mathbf{x},t)b_i^{act}(\mathbf{x},t)d\mathbf{x}. \tag{64}$$

Combing Eq. (59), Eq. (60), Eq. (63), and Eq. (64) results in the final weak form of the conservation of momentum in the Eulerian view:

$$\int_{\Omega^t} q_i\rho a_i d\mathbf{x} = \int_{\partial\Omega^t} q_i t_i ds(\mathbf{x}) - \int_{\Omega^t} q_{i,k}\sigma_{ik}^{cm}d\mathbf{x} + \int_{\Omega^t} q_i\rho b_i^{bd}d\mathbf{x} + \int_{\Omega^t} q_i\rho b_i^{act}d\mathbf{x}. \tag{65}$$

References

[1] Zhangyang Gao, Cheng Tan, Lirong Wu, and Stan Z Li. Simvp: Simpler yet better video prediction. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 3170–3180, 2022. 2

[2] Yuanming Hu, Yu Fang, Ziheng Ge, Ziyin Qu, Yixin Zhu, Andre Pradhana, and Chenfanfu Jiang. A moving least squares material point method with displacement discontinuity and two-way rigid body coupling. *ACM Transactions on Graphics (TOG)*, 37(4):1–14, 2018. 7

[3] Chenfanfu Jiang, Craig Schroeder, Andrew Selle, Joseph Teran, and Alexey Stomakhin. The affine particle-in-cell method. *ACM Transactions on Graphics (TOG)*, 34(4):1–10, 2015. 7

[4] Chenfanfu Jiang, Craig Schroeder, Joseph Teran, Alexey Stomakhin, and Andrew Selle. The material point method for simulating continuum materials. In *Acm siggraph 2016 courses*, pages 1–52. 2016. 6, 7, 9

[5] Xiaojie Jin, Huaxin Xiao, Xiaohui Shen, Jimei Yang, Zhe Lin, Yunpeng Chen, Zequn Jie, Jiashi Feng, and Shuicheng Yan. Predicting scene parsing and motion dynamics in the future. *Advances in neural information processing systems*, 30, 2017. 2

[6] A NSAV Marana, Sergio A Velastin, L da F Costa, and RA Lotufo. Automatic estimation of crowd density using texture. *Safety Science*, 28(3):165–175, 1998. 2

[7] Andre Pradhana Tampubolon, Theodore Gast, Gergely Klár, Chuyuan Fu, Joseph Teran, Chenfanfu Jiang, and Ken Museth. Multi-species simulation of porous sand and water mixtures. *ACM Transactions on Graphics (TOG)*, 36(4): 1–11, 2017. 2, 4

[8] Cheng Tan, Zhangyang Gao, Lirong Wu, Yongjie Xu, Jun Xia, Siyuan Li, and Stan Z Li. Temporal attention unit: Towards efficient spatiotemporal predictive learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 18770–18782, 2023. 2

[9] Roger Temam. *Navier-Stokes equations: theory and numerical analysis*. American Mathematical Soc., 2001. 7

[10] Adrien Treuille, Seth Cooper, and Zoran Popović. Continuum crowds. *ACM Transactions on Graphics (TOG)*, 25(3): 1160–1168, 2006. 6

[11] Benjamin Ummenhofer, Lukas Prantl, Nils Thuerey, and Vladlen Koltun. Lagrangian fluid simulation with continuous convolutions. In *International Conference on Learning Representations*, 2019. 1

[12] Yunbo Wang, Haixu Wu, Jianjin Zhang, Zhifeng Gao, Jianmin Wang, S Yu Philip, and Mingsheng Long. Predrnn: A recurrent neural network for spatiotemporal predictive learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(2):2208–2225, 2022. 2

[13] Shuang Wu, Hua Yang, Shibao Zheng, Hang Su, Yawen Fan, and Ming-Hsuan Yang. Crowd behavior analysis via curl and divergence of motion trajectories. *International Journal of Computer Vision*, 123(3):499–519, 2017. 4, 5