A COMPARISON OF AIR FLOW SIMULATION TECHNIQUES IN ARCHITECTURAL DESIGN

by

Yuanpei Zhao

A Thesis

Submitted to the Faculty of Purdue University

In Partial Fulfillment of the Requirements for the degree of

Master of Science

Department of Computer Graphics Technology

West Lafayette, Indiana

May 2021

THE PURDUE UNIVERSITY GRADUATE SCHOOL
STATEMENT OF COMMITTEE APPROVAL

Dr. Hazar Dib, Chair

Department of Computer Graphics Technology and
Department of Construction Management Technology

Dr. Tim McGraw

Department of Computer Graphics Technology

Dr. Yingjie Chen

Department of Computer Graphics Technology

Approved by:

Dr. Nicoletta Adamo-Villani

ACKNOWLEDGMENTS

Special thanks go to my graduate-committee members Prof. Hazar Dib, Prof. Tim McGraw, and Prof. Yingjie Chen for providing valuable feedback and suggestions for this project in numerous regards.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABSTRACT

The fluid simulation in computer generates realistic animations of fluids by solving Navier-Stokes equation. The methods of simulation are divided into two types. The grid-based methods and particle-based methods. The former one is wildly used for scientific computation because of its precision of simulation while the latter one is used in visual effects, games and other areas requiring real-time simulation because of the less computation time it has.

The indoor airflow simulations with HVAC system in construction design is one specific application in scientific computation and uses grid-based simulation as the general-purpose simulation does. This study addresses the problem that this kind of airflow simulations in construction design using grid-based methods are very time consuming and always need designers to do pretreatment of the building model, which takes time, money, and effort. On the other hand, the particle-based methods would have less computation time with an acceptable accuracy in indoor airflow simulations because this kind of simulation does not require very high precision.

Then this study conducts a detailed and practical comparison of different fluid simulation algorithms in both grid-based methods and particle-based ones. This study's deliverable is a comparison between particle-based and grid-based methods in indoor airflow simulations with HVAC system.

The overall methodology used to arrive at the deliverables of this study will need two parts of work. The benchmark data is gathered from a CFD software simulation using FVM with a decent grid resolution. The particle-based data will be generated by simulation algorithms over the same set of room and furniture models implemented by OpenGL and CUDA. After the benchmark FVM simulation being conducted in a CFD software, the temperature field of airflow will be measured. After simulation, the temperature field are gained on each one of 4 particle-based simulation. A comparison standard is set and data will be analyzed to get the conclusion. The result shows that in a short simulation time period, after finding a proper number of particles, the particle-based method will achieve acceptable accuracy of temperature and velocity field while using much less time.

1 PURPOSE AND PROBLEM

1.1 Introduction

The in-building airflow simulation has always been a topic in the construction design field. In the middle of 1960, the first simulation method that considered time as an independent variable appeared. Heat balance approaches were introduced in the 1970s and improved in the next several years. There were many approaches to simulate outdoor wind and indoor airflow. However, the computational fluid dynamics (CFD) approach simulating airflow with detail in single rooms (Nielsen, 1974) is the most wildly used one, which has been improved and still used now.

The CFD method is applied to many construction software for pollution detecting, temperature controlling, and many other uses. The most wildly used method of CFD method is finite volume method (FVM). FVM is a grid-based method which integrals Euler control equations for each volume and compute all volume grids separately. Many CFD software uses this method, such as FLUENT, CFX, Starccm+ and OpenFoam.

Meanwhile, many other simulation methods have not been used in this field wildly Recently, the particle-based techniques are popular in many areas, such as visual effects, games because of their simplicity and flexibility.

Smoothed particle hydrodynamics (SPH) (Monaghan, 1992) is one crucial particle-based method for fluid simulation. It is a simple implementation of the Navier-Stokes equation, which computes the force between each particle and with boundaries, then calculates the velocity and updates position of each particle. It has several improved versions like WCSPH (Becker, & Teschner, 2007) and PCISPH (Solenthaler, & Pajarola, 2009), improving the original method in various aspects. Nevertheless, all of them require a relatively small time-step to get accuracy and stability in simulation. They also need a large-scale computation in complex shapes (Harada, Koshizuka, & Kawaguchi, 2007).

Another widely used method in fluid simulation is Position-Based Dynamics (PBD) (Muller, Heidelberger, Hennix, & Ratcliff, 2007). Instead of computing forces, it updates the position of particles directly by formulating and solving a set of positional constraints. This method does not require small time-step, but the result is less accurate and difficult to adjust parameters independently (Macklin, & Muller, 2013). These two methods are not wildly used in the airflow simulation in building construction design.

1.2 The Problem

The problem addressed by this study is that indoor airflow simulations with HVAC system in construction design using grid-based FVM methods are very time consuming and always need designers to do pretreatment of the building model, which takes time, money, and effort. When doing architectural design, there are many factors to consider, like the ventilation and the temperature control, which will need to be simulated before actual construction begins. Fortunately, there is much software like AutoCAD and SolidWorks that will help designers simulate the airflow inside the building. Although all based on the FVM method, the speed they simulate an ample space is considerably slow. For example, an indoor contaminant transport simulation conducted by Wang, Dols and C, hen (2010) took 13 hours to simulate a 2 hours procedure with a one-minute time step. This simulation was relatively slow, and the time step was too large to gain a precise result. ZhaiChen, Haves, and Klems (2002) said that when using traditional FVM methods to simulation, “the conjugate heat transfer method is not practical for immediate use in a design context with current computer capabilities and speed.” Zhai and Chen (2004) combined different methods in building energy simulation, but “the long computing time restricts the applicability of the coupling program for practical design purpose.”

The need of a faster method for indoor airflow simulation is growing these days. Cao (2019) concluded that “For online control of ventilation, a ‘faster-than-real-time’ prediction is expected based on multi-inputs, i.e., ventilation modes, air change per hours (ACH) and pollutant sources etc.” Current speed of FVM methods apparently cannot meet this requirement.

Another disadvantage of FVMs is that their input of the building must be in grids that have a completed Domain/Boundary namespace. With the increasing scale of the modern buildings and each one’s complexity, Millions of grids need to be generated to achieve a FVM simulation, which always needs the help of a server to compute in a reasonable time. So, the pretreatment to create grids and simulations based on grids takes time and money in a word.

1.3 Significance

Airflow simulation is very time-consuming for the construction designers. Because FVM-based algorithms used currently often need complex computation. Particle-based algorithms like SPH and PBD simplify the calculation, so that is much faster than FVM-based ones. The complexity of CFD-based algorithms is a necessary cost to achieve accuracy when simulating complex functionality like pump and sewer systems. However, when applying to a pure environment like air flow in rooms, whether the difference of accuracy between FVMs and new particle-based methods is significant needs to be tested.

Another disadvantage of FVMs mentioned before is the need of grid-generation pretreatment. This will require designers to do the pretreatment of the building model before they can do the simulation, which costs time and money. Different software often requires different file format like *.stl or *.msh, which makes it more complex to do pretreatment. On the contrary, particle-based fluid simulation methods are grid-free, which means they do not need the environment model to grid. This is especially convenient for models generated by the point cloud.

1.4 The Purpose

The purpose of this study is to conduct a detailed and practical comparison of different fluid simulation algorithm of their accuracy and speed inside several rooms in a building to show whether the particle-based methods of the indoor airflow simulation will be faster than FVMs while remaining acceptable accuracy. If a conclusion could be reached, this result of comparison will simplify the future work of building designers while trying to choose an algorithm to simulate the air flow with indoor environment.

For the algorithms that has not been applied wildly in air flow simulation in construction building design, while they are wildly used in many other fields, the comparison these algorithms and traditional FVM-based algorithms is lacked. SPH is widely used in many software applications like AUTODESK CFD, NEUTRINO by Centroid Lab Inc. and so on. PBD is also used in many fields, for example, FleX, one of a particle-based simulation technique of Nvidia is based on PBD methods (Macklin, Muller, Chentanez, & Kim, 2014). There are some comparison work between PCISPH and WCSPH done already (Shadloo, Zainali, Yildiz, & Suleman, 2011) and some attempts to combine these methods, like combining PCISPH and WCSPH (Raveendran, Wojtan,

& Turk, 2011), using PBD methods in SPH methods (Shao, Liao, & Zhang, 2017) and so on. But the comparison is from overall aspect and is not specifically focus on air flow simulation in buildings.

1.5 Research Questions

The questions need to be answered in this study to draw a conclusion associated with the purpose are as follows:

Research Question 1: Could particle-based method achieve similar result pattern as grid-based method?

Research Question 2: If the satisfactory result is achieved, does particle-based method spend less time than grid-based method?

This study will deliver a comparison of different particle-based fluid simulation algorithms over the benchmark of FVM. A conclusion of the advantage and disadvantage of FVM-based algorithms and particle-based algorithms will be made based on the accuracy the model generated and the speed of the algorithm measured in frame per second.

1.6 Assumptions

For this project, several assumptions will be made. The performance between algorithms stays the same among different operating systems if the hardware is the same. The relative performance between algorithms remains the same among different graphics cards.

1.7 Delimitations

- This project focuses on the simulation method, only excluding rendering methods.
- This project will be operated on Windows 10 system and RTX 2080 graphics card.
- The algorithms that will be compared are particle-based methods and mainly SPH methods.
- This project will only simulate the air flow condition in different sets of one single room, which is a relatively closed and simple environment.

1.8 Limitations

- Due to the difference of grid-based and particle-based algorithms, the airflow simulation will be slightly different, which introduces irrelevant variables.
- Due to the restriction of the funding of the experiment, the real-world experiment is not included. The high cell resolution grid-based method result will be used as benchmark instead.
- Due to the restriction of SPH method, the time step needs to be small, so the total simulation time is relatively short.
- Due to personal ability, the models implemented will be simplified.
- This study will not conduct a real-world experiment. Instead, this study will use CFD software to do FVM simulation under a decent grid resolution and use the simulation result as benchmark.

1.9 Definitions

The relevant concepts used are listed below.

computational fluid dynamics (CFD) - A branch of fluid mechanics that uses numerical analysis and data structures to analyze and solve problems that involve fluid flows (Thomson, 1973).

Navier-Stokes equation - A set of partial differential equations which describe the motion of viscous fluid substances. "Conservation of momentum, mass, and energy are expressed by Navier-Stokes equations mathematically." (McLean, 2013)

flow field – "A flow field consists of a three-dimensional. sample space that returns a vector at every point, indicating an attraction toward objects of interest or repulsion away from objects to be avoided" (Alexander, 2006).

parallel computing - The process of executing a significant issue as many smaller problems simultaneously by multiple processors. The most commonly used ways in computer graphics are CUDA and compute shader.

CUDA - An environment for creating high-performance GPU-accelerated applications developed by NVIDIA.

compute shader - A shader stage in the graphics pipeline that is used for parallel computing in GPU.

point cloud – One of laser scanning technology that is used to generate accurate digital reconstructions in different formats. (Barazzetti, 2018)

grid-based fluids - One type of fluid simulation that uses the grid as a basic unit. Every sample of the fluid is fixed in a grid, and the information is passing through it. This method is less accurate and hard to use parallel computing to accelerate.

particle-based fluids - One type of fluid simulation that regards fluid as many small particles. Particles carry data samples and travel with the flow. It is the most commonly used type of simulation nowadays.

smoothed-particle hydrodynamics (SPH) - One crucial particle-based method for fluid simulation designed to model compressible flow. “It computes the force between each particle and with boundaries, then calculates the velocity and updates position of each particle. It has several improved versions.” (Solenthaler & Pajarola, 2009)

position-based dynamics (PBD) - Another widely used particle-based method in fluid simulation. Instead of computing forces, it “updates the position of particles directly by formulating and solving a set of positional constraints”. (Macklin & Muller, 2013)

The relationship among these concepts showed in Figure 1.

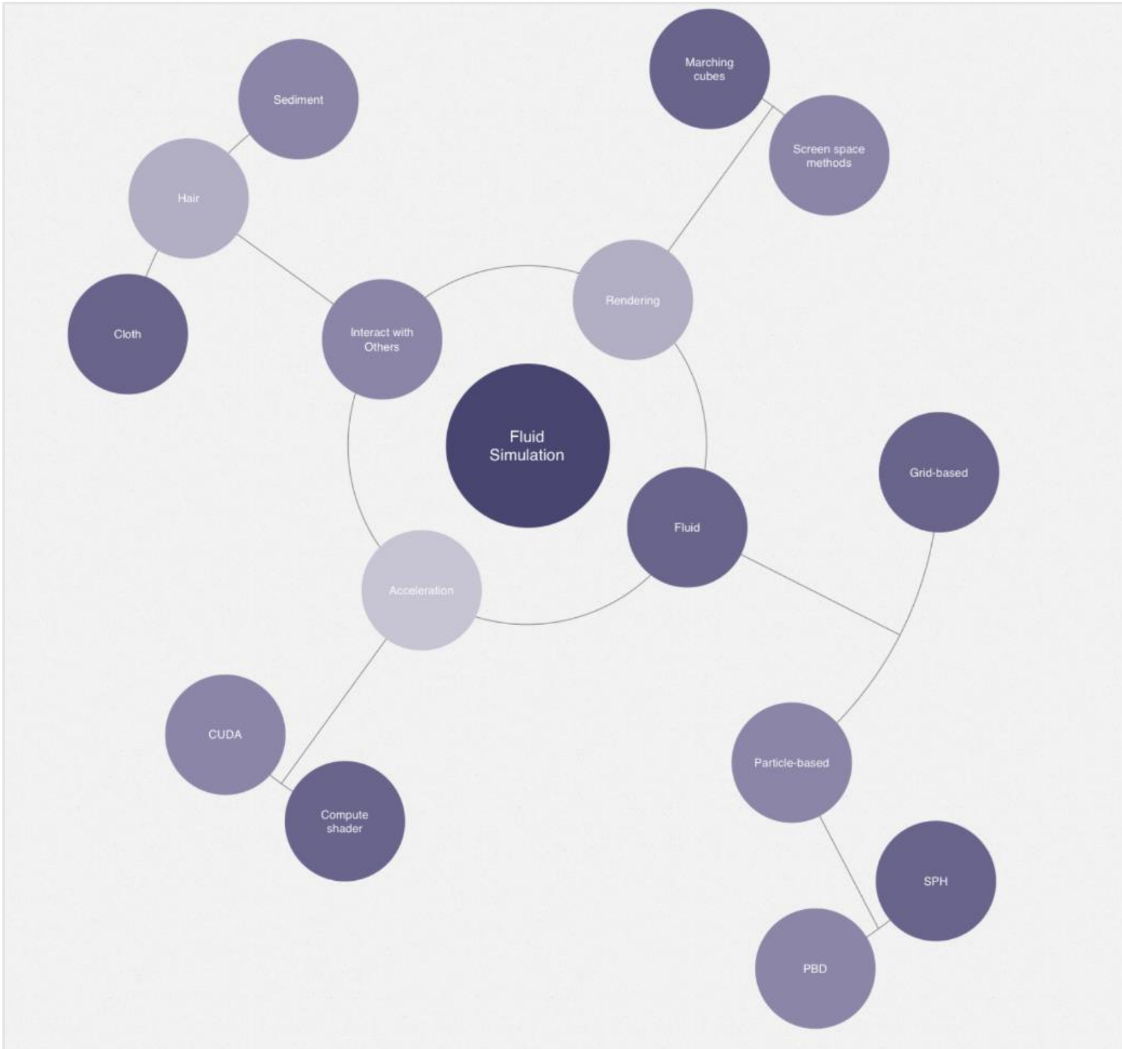

Figure 1. Concept map of definition terms

This study will only focus on the simulation part, but no rendering issues. The standard that this study wants to deliver will also consider the interaction with other objects, as mentioned in the concept map. Also, it will cover the convenience of acceleration.

2 REVIEW OF LITERATURE

2.1 Grid-based Fluid Simulation

In the 1960s of the architectural design field, estimating energy and airflow inside a building was calculated by simple math methods that were solved by hand. The first true simulation methods appeared in the middle 1960s, trying to simulate physical conditions using time as an independent variable. In the 1970s, heat balance approaches were introduced in building energy transfer simulation. Several other attempts based on this were introduced (Lebrun, 1982), while computational fluid dynamics (CFD) approaches were applying to simulate more detailed airflow in a closed single room (Nielsen, 1974).

CFD has been wildly used in the room airflow simulation for several decades. Whittle (1986), Nielsen (1989), and Jones and Whittle (1992) provide a thorough review of the applications. Morrison (2000) analyzed from user perspectives and concluded that "Usage of simulation by the design professions is growing." In energy-efficient, comfort, health aspects.

Simulating with a grid is like using an array of fixed windows. For each time, data are recorded and stored in those windows. Physical quantities are gained and calculated at each grid cell from a fixed point of view. The definition of each data representation could be divided into two subclasses: vertex-centered grid and face-centered grid. For the face-centered gird, it is also called marker-and-cell (MAC) grid, which is the name of a computational fluid dynamics technique.

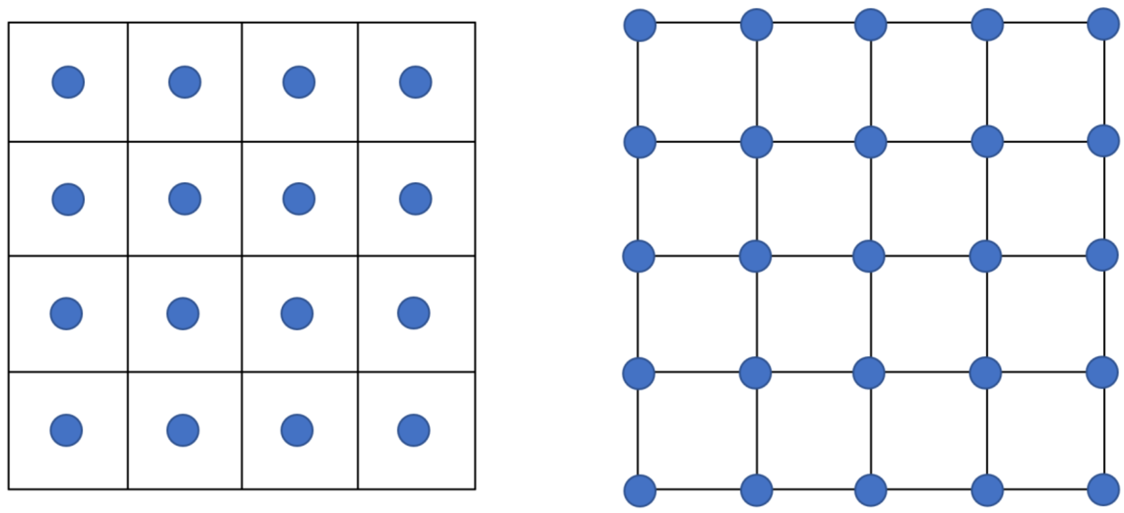

Figure 2. Face-centered grid(left) and vertex-centered grid(right)

The main components of grid-based simulation methods are external forces, viscosity, pressure gradient, and advection. The advection is a process that makes materials or quantities transfer with flow. In one grid-based simulation, each step takes a velocity field as input and modify it, then write the new velocity field as output. This is often used in single component simulation.

Nowadays, many fluid simulation software uses grid-based methods, such as FLUENT, CFX, Starccm+ and OpenFOAM. In this study, we will use OpenFOAM to simulate a grid-based method, which is buoyantSimpleFoam solver.

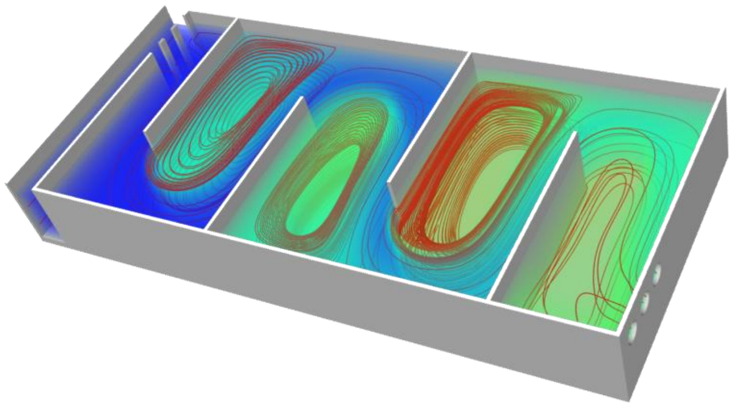

Figure 3. Example of CFD-used grid-based method simulation

OpenFOAM is mainly a C++ library, which is used primarily to create executables. The applications have two categories: solvers and utilities. The former ones are designed to solve specific problems in continuum mechanics, and the latter are designed to perform tasks that involve data manipulation. New solvers and utilities can be created by its users with some pre-requisite knowledge of the underlying method, physics and programming techniques involved.

The buoyantSimpleFoam solver is one of those solvers, which is a steady-state solver for buoyant, turbulent flow of compressible fluids, including radiation, for ventilation and heat-transfer. The domain equations of this solver are described below.

The mass continuity equation is given by this equation.

$$\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho u) = 0 \tag{1}$$

Where u is the velocity and ρ is the density. The time derivative is omitted for steady state solver.

The momentum conservation equation is given by the following equation.

$$\frac{\partial (\rho u)}{\partial t} + \nabla \cdot (\rho u u) = -\nabla p + \rho g + \nabla \cdot \big(2\mu D(u)\big) - \nabla(\frac{2}{3}\mu(\nabla \cdot u)) \tag{2}$$

$$D(u) = \frac{1}{2}(\nabla u + (\nabla u)^T) \tag{3}$$

Where p is the static pressure field and g is the gravity, and μ is the viscosity.

The internal energy equation is given by the following equation.

$$\frac{\partial(\rho e)}{\partial t} + \nabla \cdot (\rho u e) + \frac{\partial(\rho K)}{\partial t} + \nabla \cdot (\rho u K) + +\nabla \cdot (\rho u) = \nabla \cdot (\alpha \nabla \mathrm{e}) + \rho u g \tag{4}$$

$$K \equiv \frac{|u|^2}{2} \tag{5}$$

Where K is kinetic energy per unit mass, and the enthalpy per unit mass h is the sum of the internal energy per unit mass e and the kinematic pressure.

2.2 Particle-based Fluid Simulation

As for the particle-based fluid simulation methods, several commonly used methods are reviewed. The first is smoothed-particle hydrodynamics (SPH) methods. Monaghan (1992) firstly suggested in the paper to use the Navier-Stokes equation into the fluid simulation in computer graphics. After that, several minor improvements were made but were all simple implementations of the Navier-Stokes equation, which computes the force between each particle and with boundaries, then calculates the velocity and updates position of each particle.

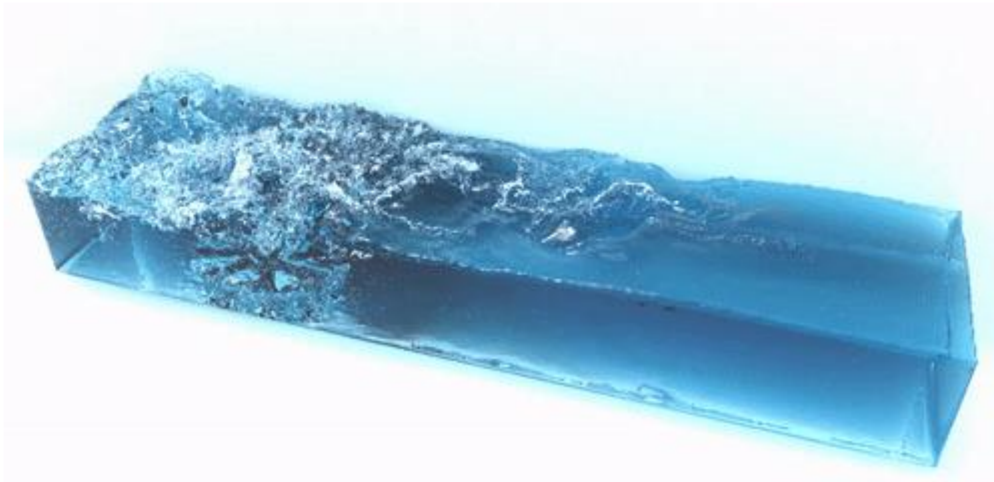

Figure 4. Example of particle-based simulation in rendering industry

The SPH method also try to solve the fluid equation, but it ignores the advection in grid-based method. The equation then could represented as following.

$$a = g - \frac{\nabla p}{\rho} + \mu \nabla^2 u \tag{6}$$

$$\nabla \cdot u = 0 \tag{7}$$

Then each force could be calculated in a discrete way, which will be discussed in next chapter.

Becker and Teschner (2007) gave an essential improvement of SPH, which is weakly compressible SPH for free surface flows (WCSPH). This method tried to solve the compressible problem in water simulation to generate a more realistic pattern of fluids. Another essential improvement is Predictive-corrective incompressible SPH (PCISPH) developed by Solenthaler and Pajarola (2009). This method predicted the particle position in the next frame based on the density of particles. It then revised this new position in an iteration until the particle density is evenly distributed. This method made the fluid more incompressible, but they needed a relatively small time-step to guarantee stability.

Figure 5. Example of Nvidia FleX

SPH is widely used in many software. Position-Based Dynamics (PBD), another particle-based fluid simulation method developed by Muller et al. (2007), is also used in many fields, for example, FleX, one of a particle-based simulation technique of Nvidia is based on PBD methods (Macklin et al., 2014). This method “updates the position of particles directly by formulating and solving a set of positional constraints” (Macklin et al., 2014) instead of computing forces as SPH methods did.

There is some comparison work between PCISPH and WCSPH done, and some attempts to combine these methods. Harada et al. (2007) analyzed that all of the SPH methods require a relatively small time-step to get accuracy and stability in simulation. They also need a large-scale computation in complex shapes.

Raveendran et al. (2011) tried to combine PCISPH and WCSPH. Shao et al. (2017) tried to use PBD methods in SPH methods. However, the comparison is from the overall aspect and is not specifically focus on airflow simulation in buildings.

2.3 Heat Transfer in Fluid Simulation

For the heat transfer simulation, Cleary (1998) used thermal energy gradient to simulate heat transfer and showed thermal boundaries. The natural convection was also simulated using the Boussinesq approximation by changing gravity force in SPH to be relevant to temperature.

Chen et al. (1999) improved the boundary value problems in heat conduction. Jiang and Sousa (2006) improved the model for ballistic-diffusive heat conduction. Rook et al. (2007) proposed a model to apply Crank-Nicolson implicit time integration technique into SPH heat transfer simulation.

The heat transfer equations are described as the following.

$$U_i = \Delta t \sum_j \frac{4m_j}{\rho_i \rho_j} \frac{k_i k_j}{k_i + k_j} (T_i - T_j) \frac{(r_i - r_j)\nabla(W_{std}(pos_i) - W_{std}(r_j))}{\left|pos_i - pos_j\right|^2 + \eta^2} \tag{8}$$

$$T_i = \frac{U_i}{c_v} \tag{9}$$

3 METHODOLOGY

3.1 Introduction

The problem addressed by this study is that indoor airflow simulations with HVAC system in construction design using FVM are very time consuming and always need designers to do pretreatment of the building model, which takes time, money, and effort. Because FVM-based algorithms used currently often need complex computation. Particle-based algorithms like SPH, PCISPH, WCSPH and PBD simplify the calculation, so that is much faster than FVM-based ones.

The deliverable of this study, therefore, will be a comparison among a particle-based method including based on SPH and a grid-based method based on buoyantSimpleFoam solver over a benchmark of buoyantSimpleFoam in building construction simulation industry, focusing on the indoor airflow simulation with HVAC system. This study will do a detailed comparison between both kinds of fluid simulation methods under different sets of layouts and different room types.

The overall methodology used to arrive at the deliverables of this study will need two parts of work. The benchmark data will be gathered from a CFD software simulation using FVM with a decent grid resolution. The particle-based data will be generated by simulation algorithms over the same set of room and furniture models implemented by OpenGL. After the benchmark grid-based simulation being conducted in a CFD software, the temperature field of airflow and velocity field of airflow will be measured. Having the same material parameters of the wall as well as relevant physical parameter of airflow, the particle-based simulation will be conducted in computer. After simulation, the temperature field and velocity field are gained on each one of fluid simulation. A comparison standard will get set and data will be analyzed to get the conclusion.

The following section will answer the research questions stated before.

3.2 Research Question

Research Question 1: Could particle-based method achieve similar result pattern as grid-based method?

Research Question 2: If the satisfactory result is achieved, does particle-based method spend less time than grid-based method?

3.3 Research Approach

This study is a quantitative research which will analyze the indoor airflow temperature accuracy of the algorithms and the speed performance of the algorithms. The data of the airflow temperature is from two sources, the FVM by OpenFOAM buoyantSimpleFoam solver and implemented particle-based SPH algorithm.

This study intends to show whether the particle-based method of the indoor airflow simulation will be faster than grid-based one while remaining acceptable accuracy. So, the speed of the algorithms will be measured as well as the accuracy. The speed will be measured in total seconds spend finishing the simulation as most algorithm comparisons do. The variable used to measure accuracy is described below.

The indoor airflow simulation with HVAC system mainly focus on velocity flow field of airflow and temperature distribution. This study mainly focuses on the temperature field of indoor airflow. So, the temperature field of the indoor airflow is chosen as a major variable to show the simulation accuracy.

The plan of this study is to use 3 different sets of room for test cases. After gaining the building model in computer, this study will use OpenFOAM to apply traditional FVMs to the room environment and simulate the temperature field of the airflow over time. Then a similar procedure will be applied over 4 different sets of particle-based algorithm with different particle number implemented by OpenGL on CPU.

The FVM and the particle-based methods are both simplification of Navier-Stokes equation and follow energy conservation law. Thus, these methods require an initial status of airflow including pressure distribution, boundary conditions, and viscosity of air. These physical parameters will be measured or inquired before the experiment. When doing simulation, the position of the wall and furniture will be the same in CFD software and OpenGL implemented ones.

Because the FVM is based on 3D grids generated during pretreatment, the resolution of the grids influences the result. So, the simulation will set a decent resolution of the grids and show the result. The result of particle-based methods depends on the number of particles. So, the particle-based simulation will be conducted under different particle number settings.

The simulation procedure is described. Then the speed of FVM will be recorded and measured in total seconds spend. Following same steps, this study will apply particle-based method to the environment and simulate the temperature field of the airflow over time. The simulation algorithms this study uses are all open source and well-known algorithms. The SPH-based methods might include WCSPH (Becker & Teschner, 2007) and PCISPH (Solenthaler & Pajarola, 2009). The PBD-based methods will include the work of Macklin & Muller (2013) which is also used in one simulation tool, FleX, by Nvidia. The speed of these methods will be recorded and measured as well.

The accuracy functions will use benchmark data generated by CFD software using FVM and experiment data generated by 4 sets of particle-based methods. Then a random sampling is conducted on temperature field data to gain several temperatures on sample points on FVM result. Then the temperatures in particle-based simulation results are inquired in sample points with same position.

The particle-based simulation results will be compared to the FVM simulation results to find the accuracy of these algorithms. The accuracy of one simulation algorithm is measured using a relative margin of error between particle-based data and FVM data.

The amount of resources needed to perform the simulation is another standard. The resources include CPU usage rate, GPU usage rate and memory usage. Other comparisons may be added during the investigation and they will be included in this section.

3.4 Parameters

All of the parameters used in this project are described below.

Table 1. Parameters used in the project

Parameter	Description	value
u	velocity	calculated
μ	dynamic viscosity	1.831e-05
p	pressure	calculated
ρ	density	calculated
g	gravity	(0, -9.8, 0)
r	Particle radius	0.005
k	conductivity	3.75e-04
c_s	Speed of sound	340
c_v	Specific heat	1004.4
U	internal energy	calculated
T	temperature	calculated
η	small parameter	1e-10
Δt	Time step	0.01
a	acceleration	calculated
pos	Position of particle	calculated

4 IMPLEMENTATION

4.1 Introduction

This chapter displays the detailed process of developing the project. First the project defines 3 different test cases for room air simulation with simplified HVAC system. For each test case, the project uses OpenFOAM to simulate a grid-based method, buoyantSimpleFoam. The block for the room is evenly divided into size of 20*20*20, 40*40*40, 60*60*60,80*80*80, and 100*100*100, in which 100*100*100 is set to be the baseline for the later precision comparison. Then the project uses ParaView to visualize the temperature field result and extracts the data for further analysis. For particle-based method, the project uses OpenGL to simulate SPH method with particle size of 10,000, 20,000 and 30,000. The temperature field result is visualized directly on the screen using drawcalls. Finally, the visualized results are compared, and the relevant precision is calculated and analyzed.

4.2 Test Cases Definition

The project will test two different fluid simulation methods (grid-based and particle-based) on three different room models with simplified inlets and outlets. These three room-models have different geometry shape. These test cases are shown below.

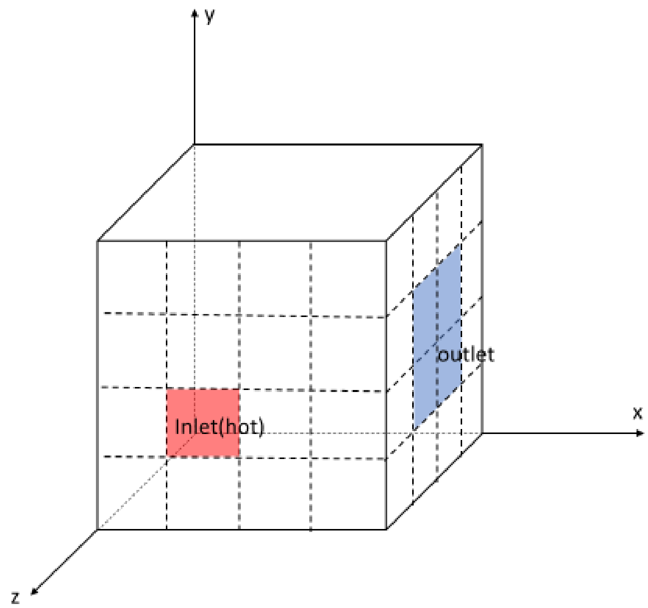

Figure 6. Test case 1

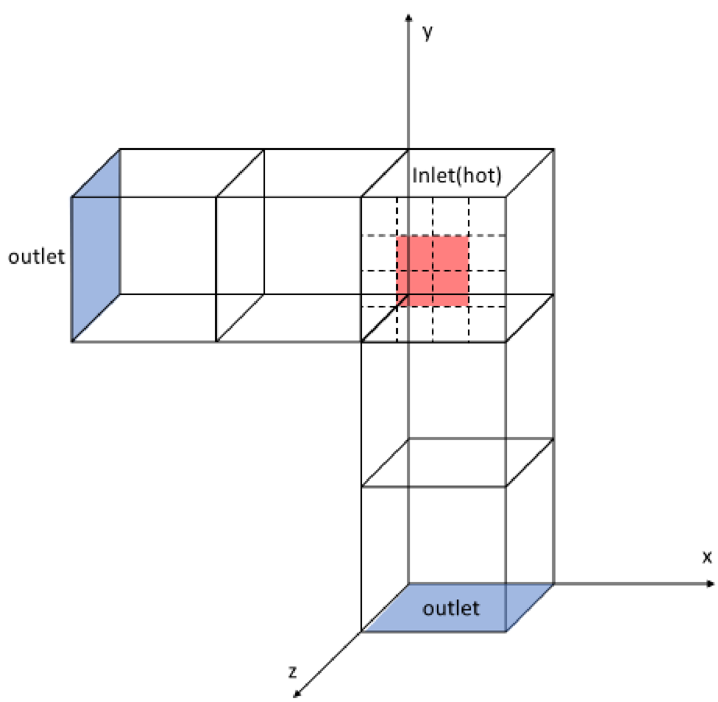

Figure 7. Test case 2

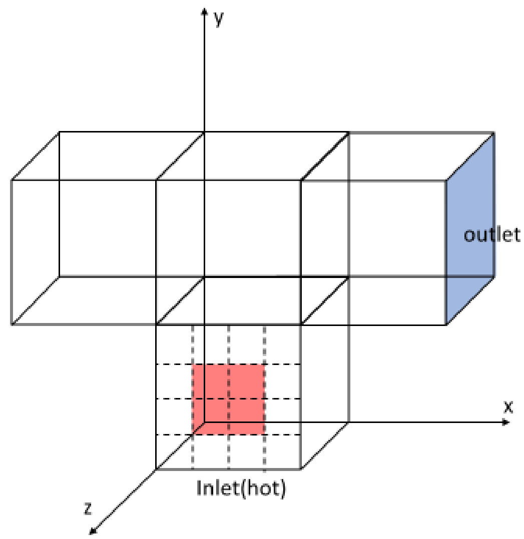

Figure 8. Test case 3

The cubes in each figure represent the room space. The size of each cube is 1m * 1m * 1m. The neighboring cubes are connected without barrier like walls. Test case 1 is a simple cube space; test case 2 is an L-shaped space; test case 3 is a T-shaped space. Each room has one inlet and one or two outlets for airflow to simulate HVAC system.

The inlet sets a fixed value for air temperature and velocity. The outlet sets a fixed value for air temperature and reference air pressure. The configuration table is as following.

Table 2. Configuration of test cases

Test Case	Type	Temperature(°F)	Pressure($g^{-1}m^{-2}$)	Velocity(ms^{-1})
1	Inlet	307.75	calculated	(0, 0, -0.05)
	Outlet	293	1e5	calculated
2	Inlet	307.75	calculated	(0, 0, -0.05)
	Outlet1	293	1e5	calculated
	Outlet2	293	1e5	calculated
3	Inlet	307.75	calculated	(0, 0, -0.05)
	Outlet	293	1e5	calculated

4.3 Grid-based Method Simulation

This project uses OpenFOAM to simulate grid-based method. Because the simulation needs to consider heat transfer and buoyant driven flow, *buoyantSimpleFoam* solver is an ideal choice. This solver is a steady-state solver for buoyant, turbulent flow of compressible fluids, including radiation, for ventilation and heat-transfer.

First, the project sets the block mesh configuration. This is to subdivide the blocks represented before in three different room sets into smaller cells for grid-based solver to calculate. For each sets of room, each block is subdivided into smaller cells with number of 20*20*20, 40*40*40, 60*60*60, 80*80*80, and 100*100*100. The 100*100*100 setting is set for the standard result and others will refer to this result and get the relative precision, which will be further discussed later.

Second, the project sets the boundary condition at start time. It includes pressure, temperature, velocity and other parameters relevant to *buoyantSimpleFoam* turbulence model, which will be included in appendix. The boundary condition of temperature and velocity are set based on the settings in table 2.

After setting these configuration files, we build the mesh by running blockMesh command and run the simulation by running buoyantSimpleFoam command. The time step is set to 0.01 second for all simulation. The total simulation time period is 10 seconds. The results are recorded every 50 iterations of simulation, which means a 0.5 second time interval.

After the simulation, we get the temperation, pressure, and velocity for each cell during 10 seconds of simulation. For the comparison and visualization of data, the project chooses ParaView to show and treat the data. For the visualization, we set the color relevant to and vectors representing velocity field. For data analysis, we use spreadsheet to extract the raw data and then select sample points to compare. The window of paraView visualization and spreadsheet looks like the following.

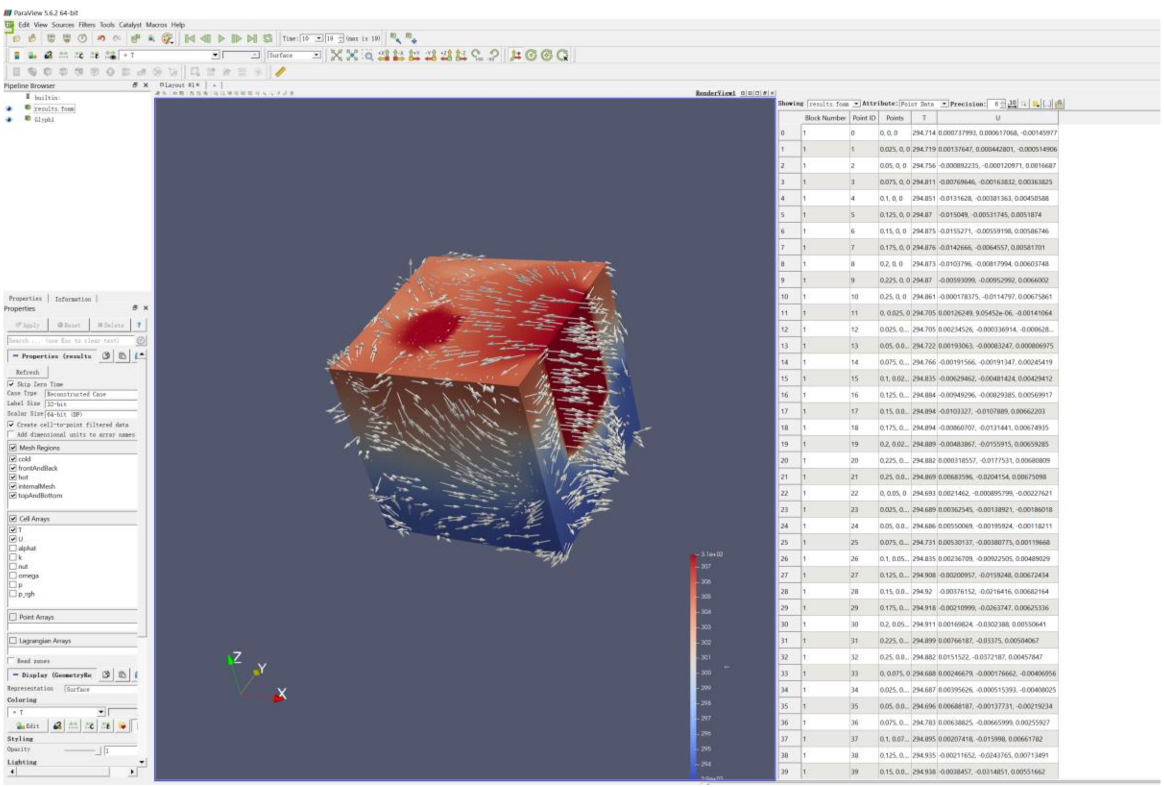

Figure X.Figure 9. Working window of ParaView

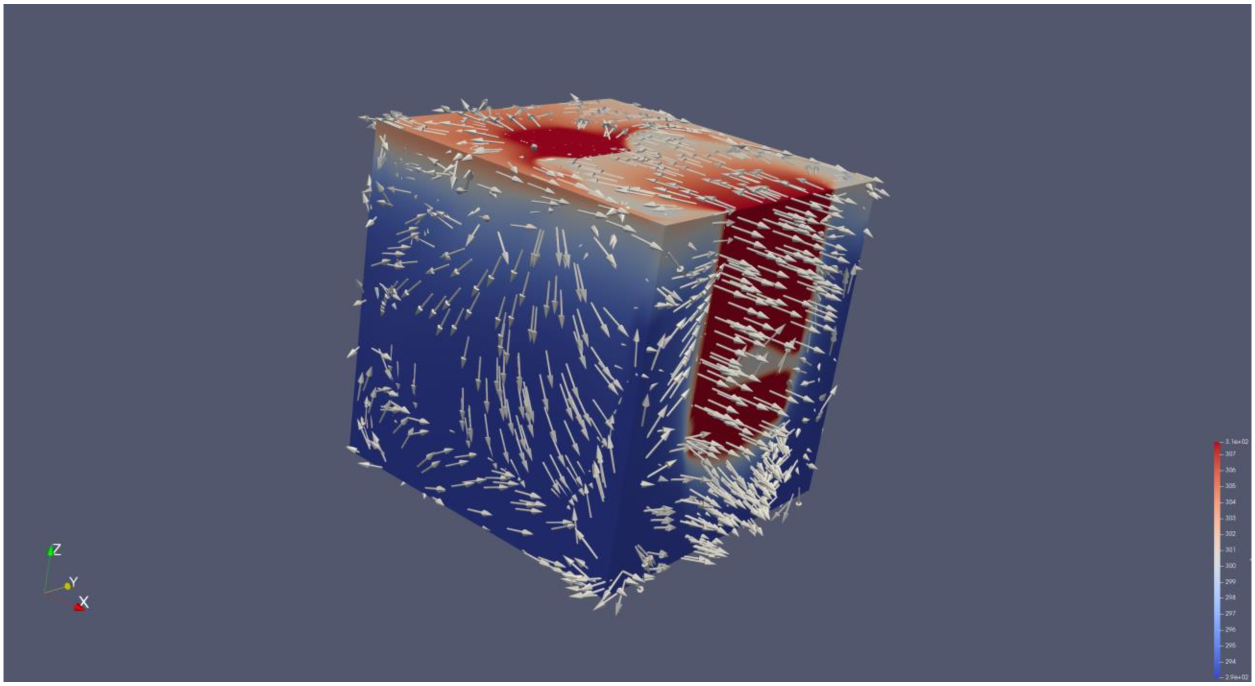

Figure 10. Visualization result for test case 1 in 100*100*100 subdivision

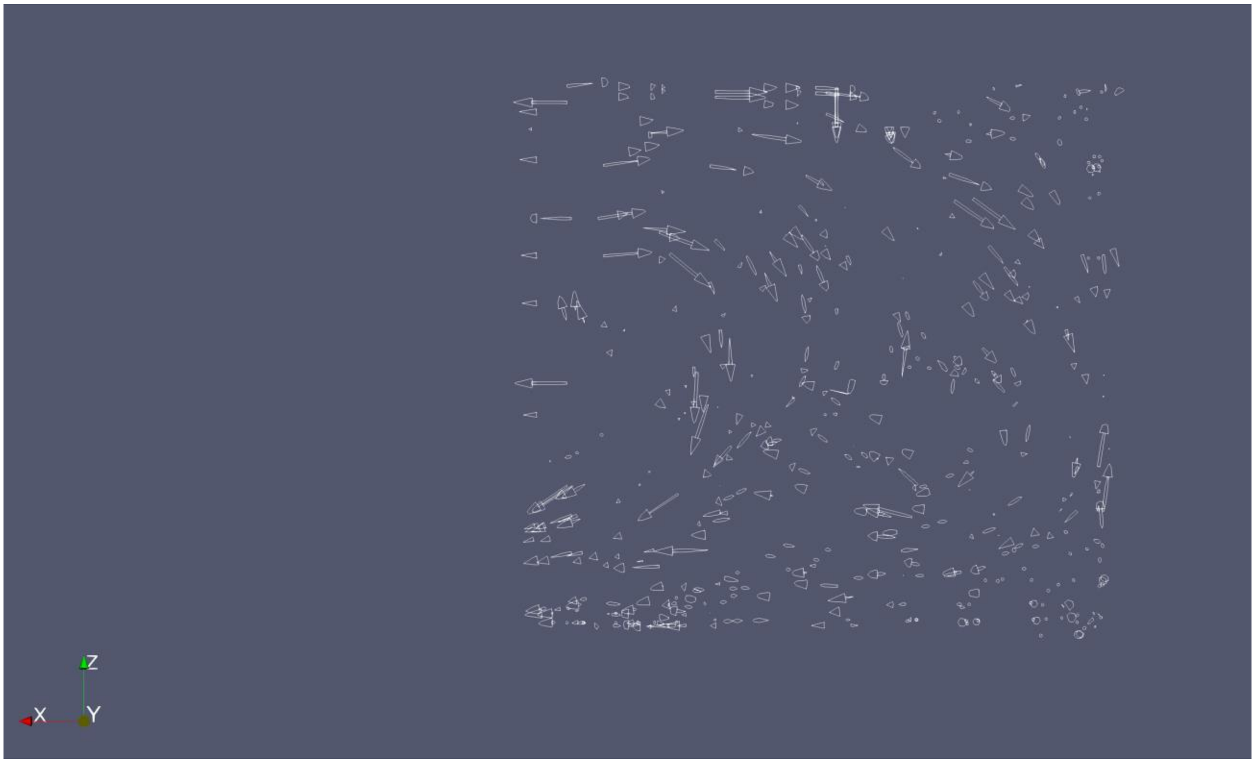

Figure 11. Velocity field visualization

4.4 SPH Simulation

The project uses OpenGL 4.3 to simulate the SPH method. The algorithm is implemented on CPU using 3-level octree. The basic SPH simulation method without heat transfer or buoyant is represented as following.

We represent fluid as many particles, and each particle represents a small fraction of the volume. The method uses a smoothed distribution of particles' physical quantities like pressure field. This smoothing function is called kernel function. This project uses poly6 kernel function, like the following, where r is the distance for a pair of particles, and h is the kernel radius.

$$W_{std}(r) = \frac{315}{64\pi h^3}\begin{cases}(1-\frac{r^2}{h^2})^3 & 0 \le r \le h \\ 0 & otherwise\end{cases} \quad (10)$$

The calculation uses 3 passes. The first pass calculated the density of each particle ρ_i and pressure p_i. The equation is like the following.

$$\rho_i = m\frac{315}{64\pi h^9}\sum_j (h^2 - |pos_i - pos_j|^2)^3, \qquad i \neq j \quad (11)$$

$$p_i = k\rho_i \quad (12)$$

Then for the second pass, the project calculated the acceleration of each particle based on the forces. Example equations for pressure and viscosity acceleration are represented as following.

$$a_i^{pressure} = \frac{-\nabla p_i}{\rho_i} = m\frac{45}{\pi h^6}\sum_j (\frac{p_i + p_j}{2\rho_i\rho_j}(h-r)^2\frac{r_i - r_j}{r}) \quad (13)$$

$$a_i^{viscosity} = \frac{f_i^{viscosity}}{\rho_i} = m\mu\frac{45}{\pi h^6}\sum_j \frac{u_j - u_i}{\rho_i\rho_j}(h - |r_i - r_j|) \quad (14)$$

Then in the final pass, the project calculated the velocity and new position for each particle.

$$v = v_{prev} + a\Delta t \tag{15}$$

$$pos = pos_{prev} + v\Delta t \tag{16}$$

This is the standard SPH simulation without heat transfer. To add heat transfer, the project adds an energy equation for each particle.

$$U_i = \Delta t \sum_j \frac{4m_j}{\rho_i \rho_j} \frac{k_i k_j}{k_i + k_j} (T_i - T_j) \frac{(r_i - r_j)\nabla(W_{std}(pos_i) - W_{std}(r_j))}{\left|pos_i - pos_j\right|^2 + \eta^2} \tag{17}$$

$$T_i = \frac{U_i}{c_v} \tag{18}$$

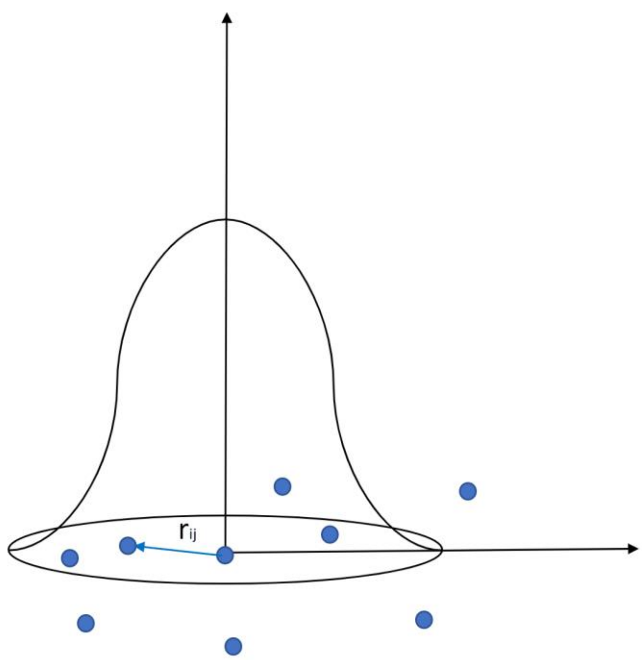

Figure 12. Kernel function

All of the calculation described before are implemented on CPU using a full 3-level octree for acceleration. Every leave node of the octree stores a linked list representing a list of particles in that block. Instead of searching every other particle, each particle just needs to search in their own block. When the particle is near the boundary (the distance to boundary is less than the kernel radius), it will then search for the neighboring block and search particles in those blocks. After calculating the position for next frame, the octree will be updated. Every particle will move to the linked list in the right block. The data structure of each particle and the full 3-level octree is described below.

```
struct particle{
  vec3   currPos;
  vec3   prevPos;
  vec3   acceleration;
  vec3   velocity;
  double pressure;
  double density;
  double energy;
  double temperature;
};
```

Figure 13. Data structure of particle

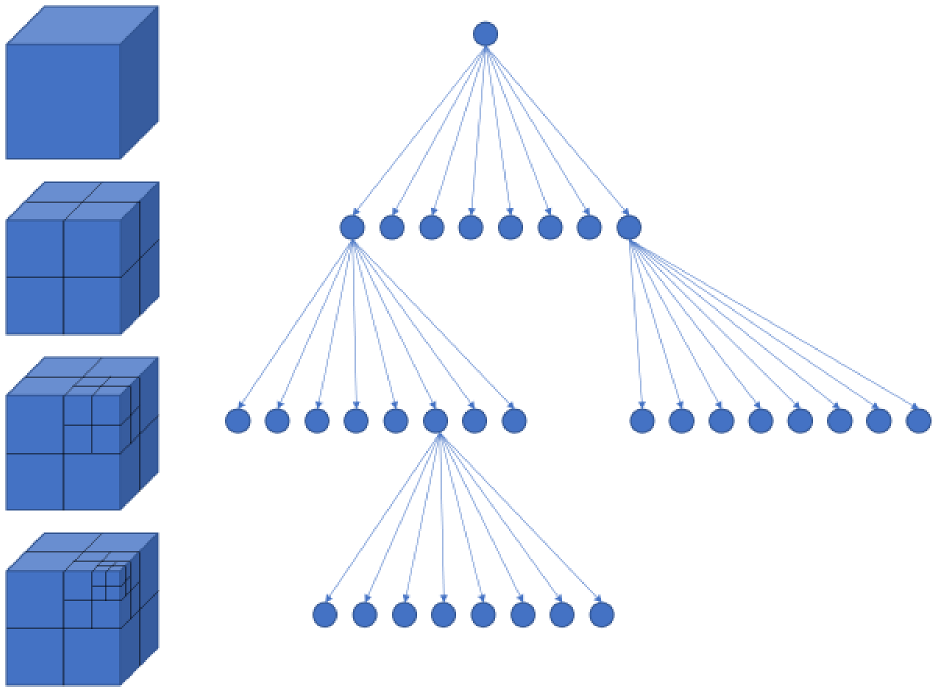

Figure 14. 3-level full octree

The visualization of temperature field is implemented by OpenGL 4.3. The position and temperature of each particle are bind to VBO. Then call the draw points command to draw each particle. In geometry shader, each particle emits four new vertices, representing a quad. Then in fragment shader, the fragment outside the particle radius will be discarded. Then the color is calculated by the following function. A final result of the particle is like the following.

```
vec3 CalcColor(float T)
{
    if(T < 300.217) return vec3((T-293.0f)/7.217f, (T-293.0f)/7.217f, 1.0f);
    else return vec3(1.0f, (307.75f - T)/7.533f, (307.75f - T)/7.533f);
}
```

Figure 15. Color calculation function

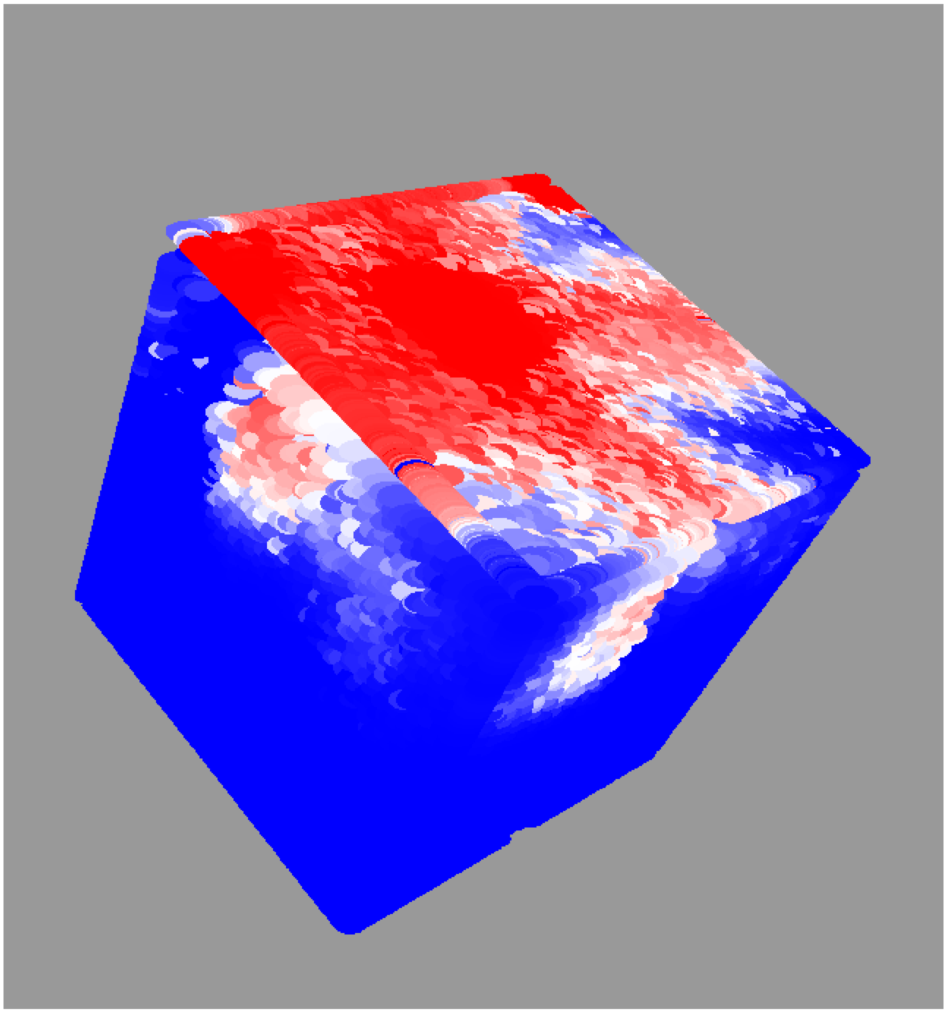

Figure 16. Example of SPH simulation for case 1

4.5 Data Treatment

After the simulation of both methods(grid-based one using OpenFOAM and particle-based one using OpenGL), we get the velocity and temperature data in the room field for every 50 time steps. For OpenFOAM, the data is extracted to .csv files using spreadsheet in ParaView. For OpenGL, it is very easy to write the relevant data to .csv files too. Then the data is read and those are near sample points are selected. We choose 10*10*10 evenly distributed sample points in every block, which means 1,000 sample points in case 1, 5,000 sample points in case 2 and 4,000 sample points in case 3 are taken. For each sample point, the average velocity and temperature value will be taken from those cells or particles within a certain radius. The pseudocode of data selection is like following.

```
WriteIn(data)

For each point in data:
        For each samplePoint in sampleList:
                If Distance(point, samplePoint) < radius:
                        samplePoint.velocity += point.velocity
                        samplePoint.temperature += point.temperature
                        samplePoint.pointSize += 1;
For each samplePoint in sampleList:
        samplePoint.velocity /= samplePoint.pointSize
        samplePoint.temperature /= samplePoint.pointSize
WriteOut(sampleData)
```

The sampled data for each test case will then be compared. The baseline data is calculated from the data in grid-based simulation with 100*100*100 cell size. For temperature, the precision of a simulation is calculated as following.

$$precision_i = \frac{|samplePoint_i.t - samplePoint100_i.t|}{maxT - minT} \tag{19}$$

$$totalPresition = \frac{1}{sampleSize}\sum_{i=1}^{sampleSize} precision_i \tag{20}$$

For velocity, the precision of a simulation is calculated as following.

$$precision_i = \cos(samplePoint_i.v, samplePoint100_i.v) = \frac{v_1 \cdot v_2}{|v_1 \cdot v_2|} \tag{21}$$

The total run time of each simulation is also recorded. The next step is to analyze the data.

5 RESULT AND ANALYSIS

5.1 Visualized result of temperature

The following is the result of gird-based method with 20, 40, 60, 80, 100 cell number and particle-based method. The result is the cross section of the room in temperature field. The first set is from test case 1, a simple cubic room with an inlet and an outlet.

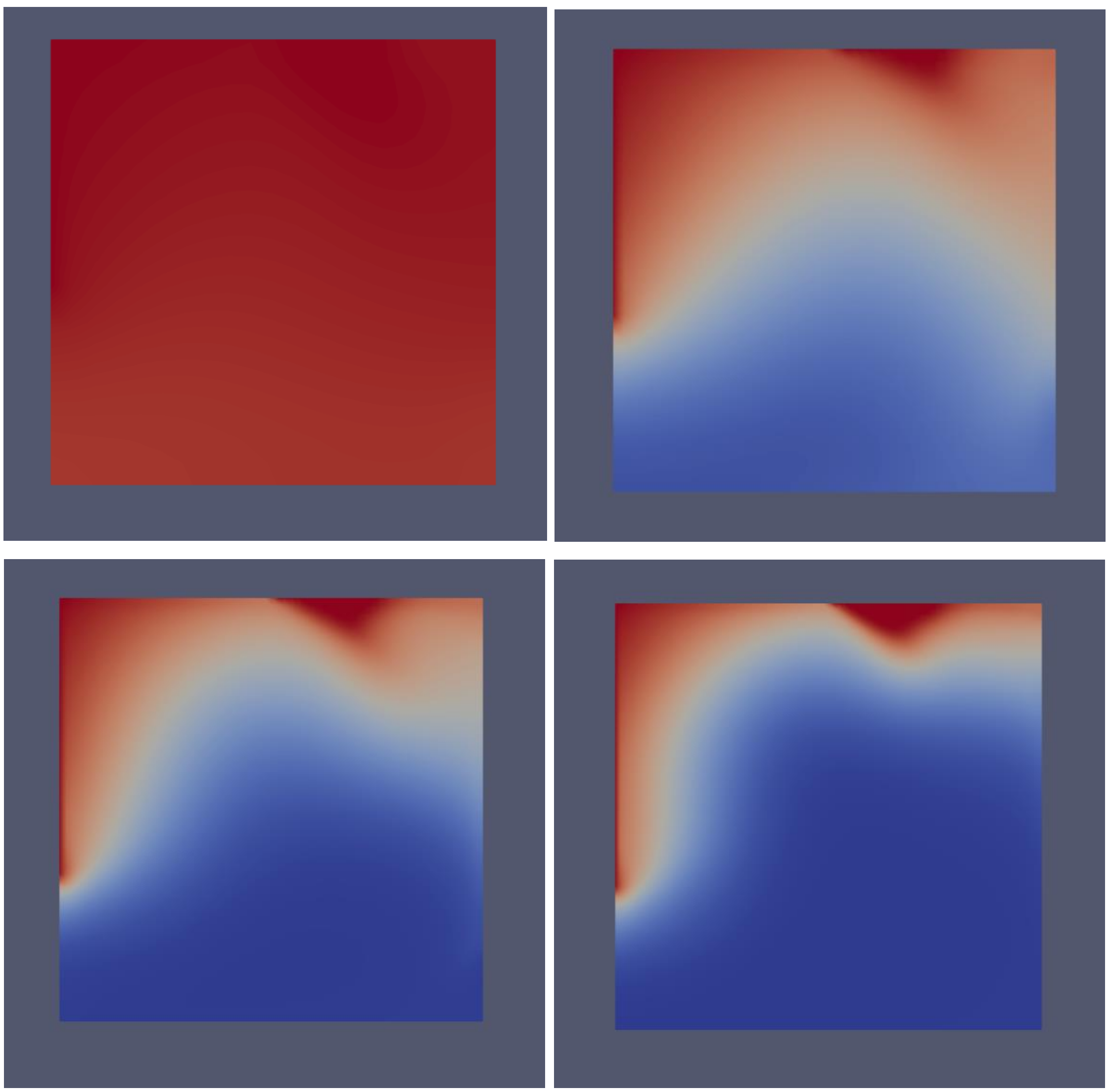

Figure 17. Temperature visualization of test case 1 with cell size of 20(top left), 40(top right), 60(bottom left), 80(bottom right)

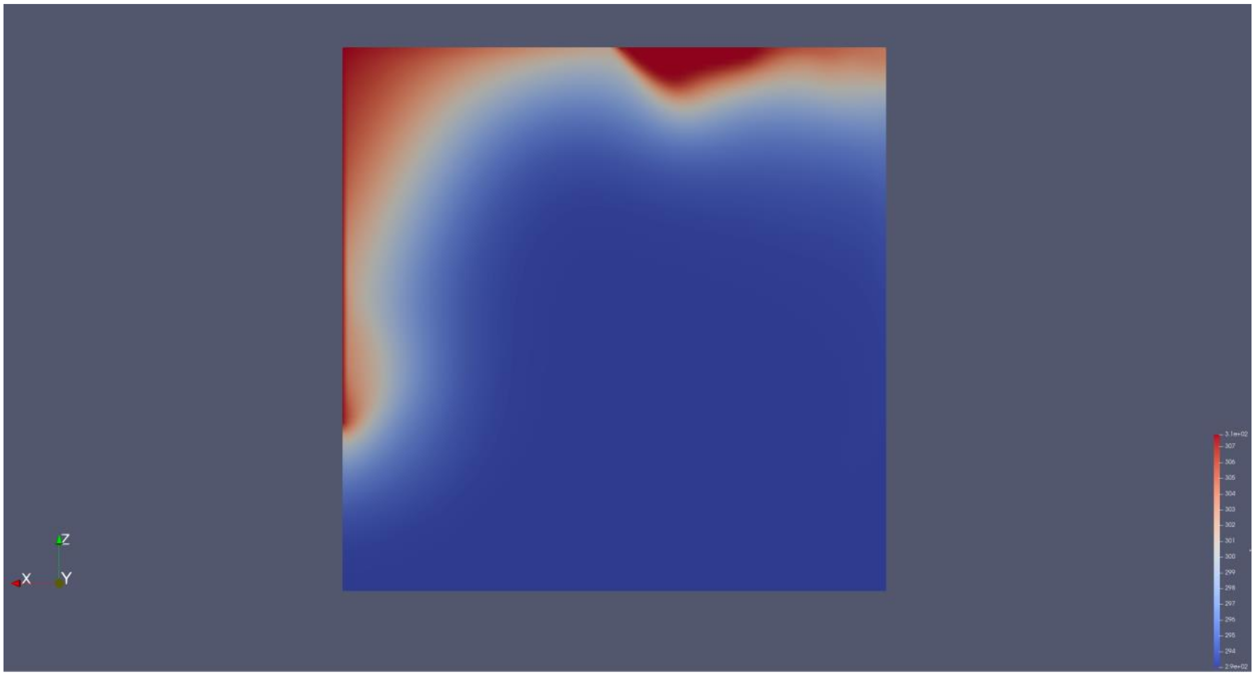

Figure 18. Temperature visualization of test case 1 with cell size of 100

For the 20*20*20 cells, we could merely recognize the overall pattern, but the heat spreads too fast so that the overall temperature is obviously higher than the baseline. For the other 3 numbers of cells, the overall pattern is clearly viewed, and with cells number increasing, the spread of heat is slower and slower.

The visualization result shows a huge change from grid resolution of 20 to 40. The study then added a set of more detailed visualization result with grid resolution of 20, 24, 28, 32, 36, 40. The result is shown in Figure 19. The accuracy of this set of experiment is also recorded and shown in next section.

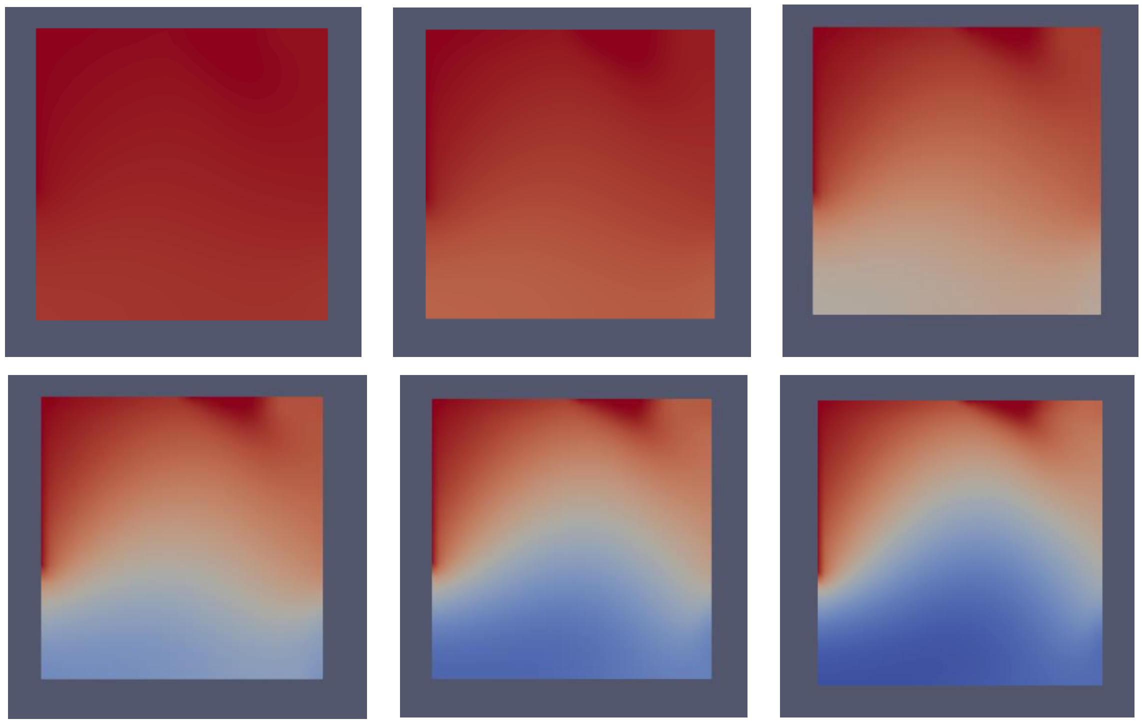

Figure 19. A more detailed temperature visualization with grid size from 20 to 40(20, 24, 28, 32, 36, 40 from top to bottom, left to right)

For the SPH method, we could also tell the same result from 4 different sets of particles. The 10,000 particles number set is totally in red and the pattern could not be recognized. The other 3 sets do have the pattern, and with particles number increasing, the spread of heat is slower and slower.

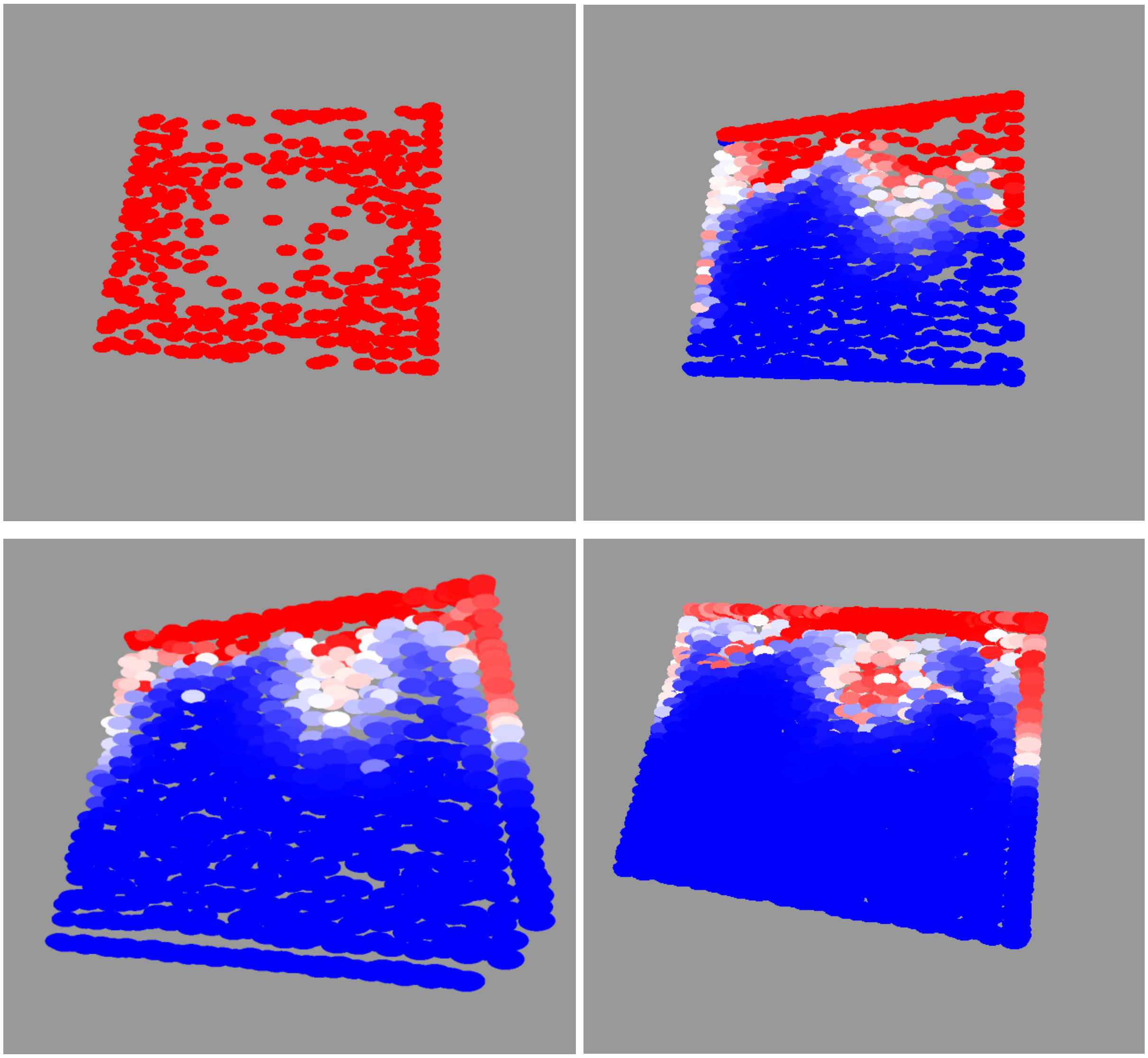

Figure 20. Temperature visualization in SPH of test case 1 with particle number of 10,000(top left), 20,000(top right), 30,000(bottom left), 40,000(bottom right)

The other 2 test cases show similar result, and the visualized figures are represented in the appendix.

5.2 Data Analysis

The next figure shows the precision of temperature and velocity for every 50 time-steps. Each line represents a simulation method. The g20 means grid-based method using 20*20 *20 cells, and the p10000 means particle-based method using 10,000 particles.

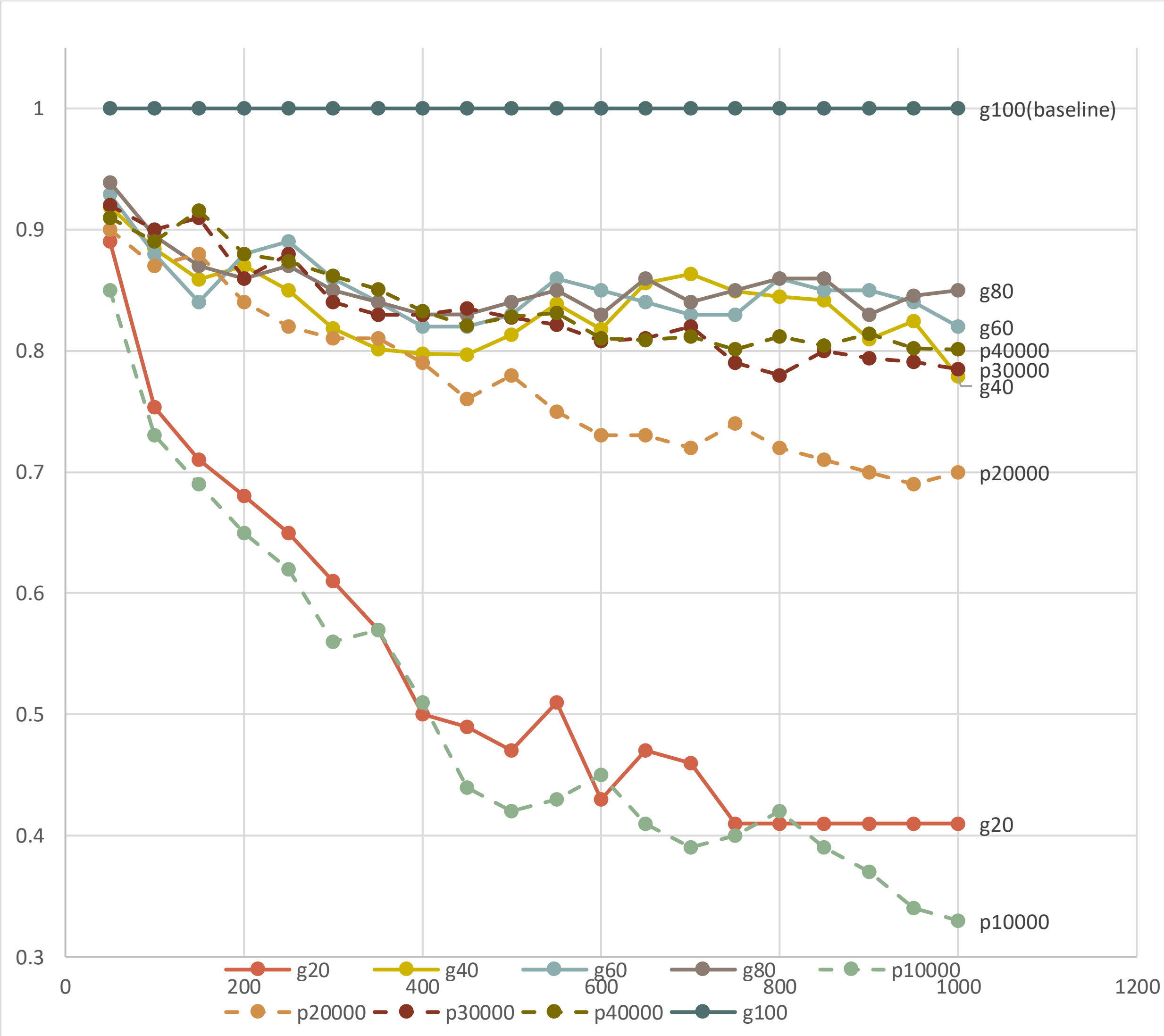

Figure 21. Temperature precision for different simulation

From Figure 21, we could see each set of simulation representing a line. And each point on the line is the precision at that time step. First, we could tell that the precision decreases with time going by in an overall trend. Second, we could tell that except g20 and p10000, every other simulation gets similar precision of pressure, which is partly because that the room is relatively large for the flowing hot air. The heat transfer only happens in a small corner of the room and other parts of room maintain the default value. Third, within each simulation method, the more cells or particles are, the higher precision will achieve.

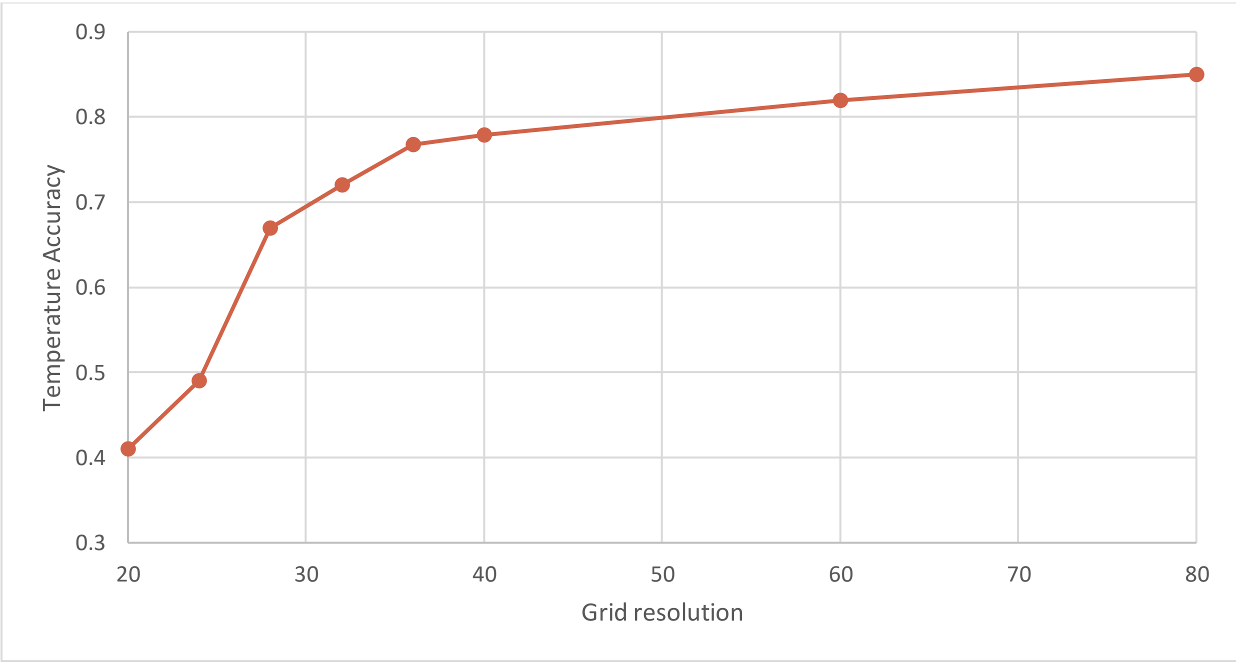

Figure 22. Temperature accuracy of grid-based method change with grid resolution

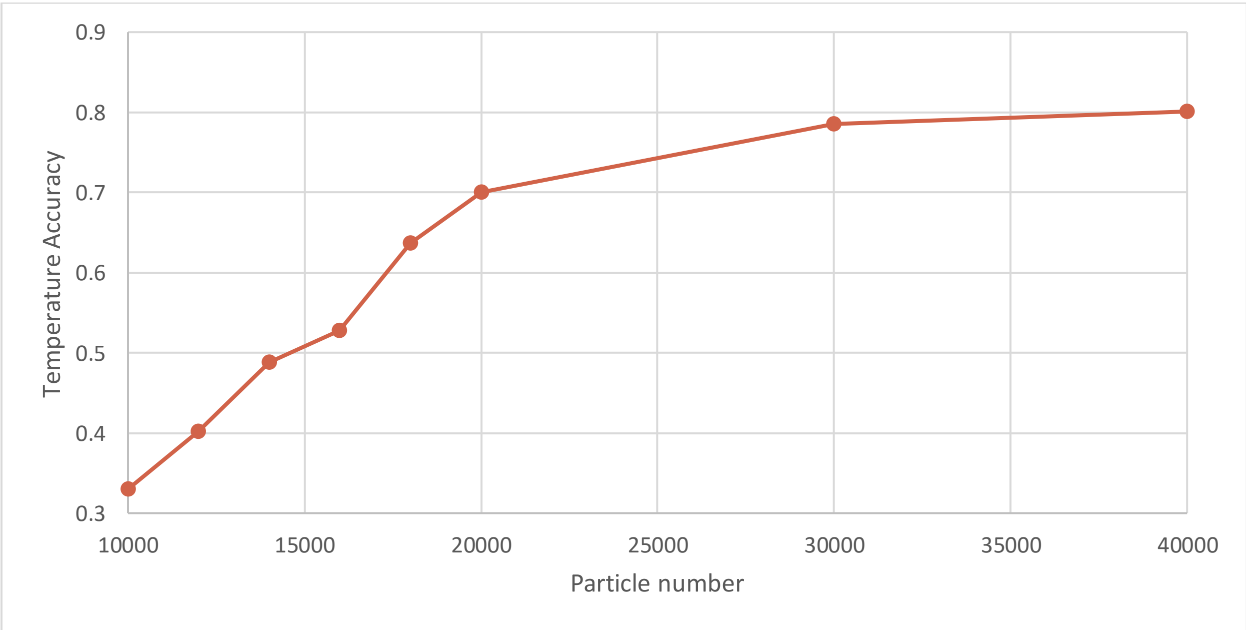

Figure 23. Temperature accuracy of particle-based method change with particle number

Figure 22 shows the accuracy of temperature increases with grid resolution increasing in grid-based method. Figure 23 shows the accuracy of temperature increases with particle number increasing in particle-based method. We could tell from figure 22 that there is a sudden increase of accuracy begin with resolution of 20 to 32. Then the increase speed gets slower when grid resolution increases after 32. From this figure we could assume grid resolution of 40 reaches an acceptable accuracy and the increase of resolution has little benefit on accuracy afterwards. For particle-based method, the curve is smooth. The increasing rate of particle number is linear, but the linear increasing of grid-resolution causes cubic increase of cell number in grid-based method. This is one of the explanations of the difference between these two charts.

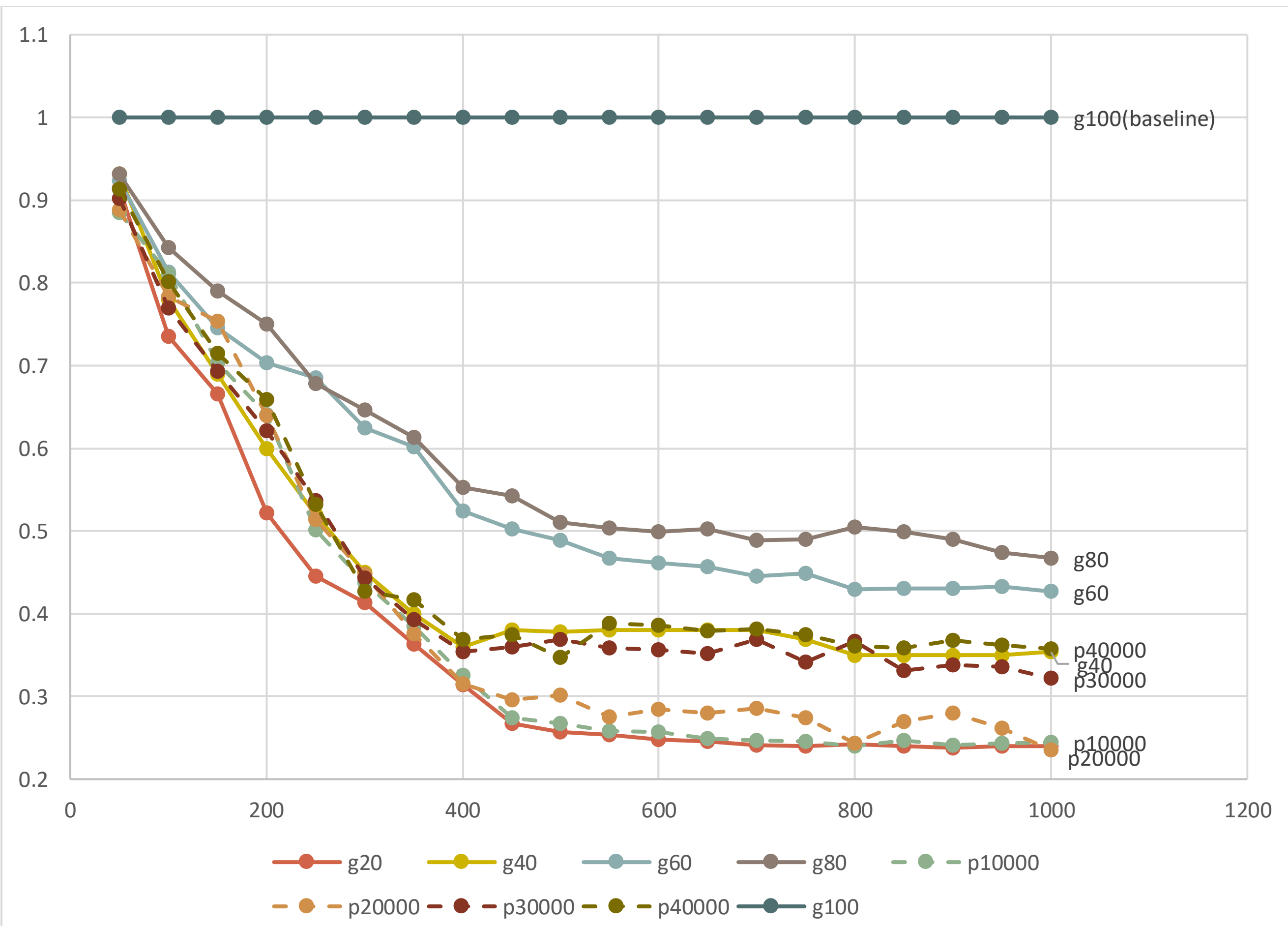

Figure 24. Velocity precision for different simulation

This chart follows the same logic as the previous one. The differences from temperature precision are from two points. First, the precision drops quickly at the beginning, but get slower during long time period. Second, the precision of simulation with different cells or particles numbers is almost evenly distributed, rather than gathering to two bunches in previous figure. From this figure, we could tell the precision of p30000 and p40000 is very near g40 method. It partly shows the ability of particle-based method to do airflow simulation in HVAC system with satisfactory result.

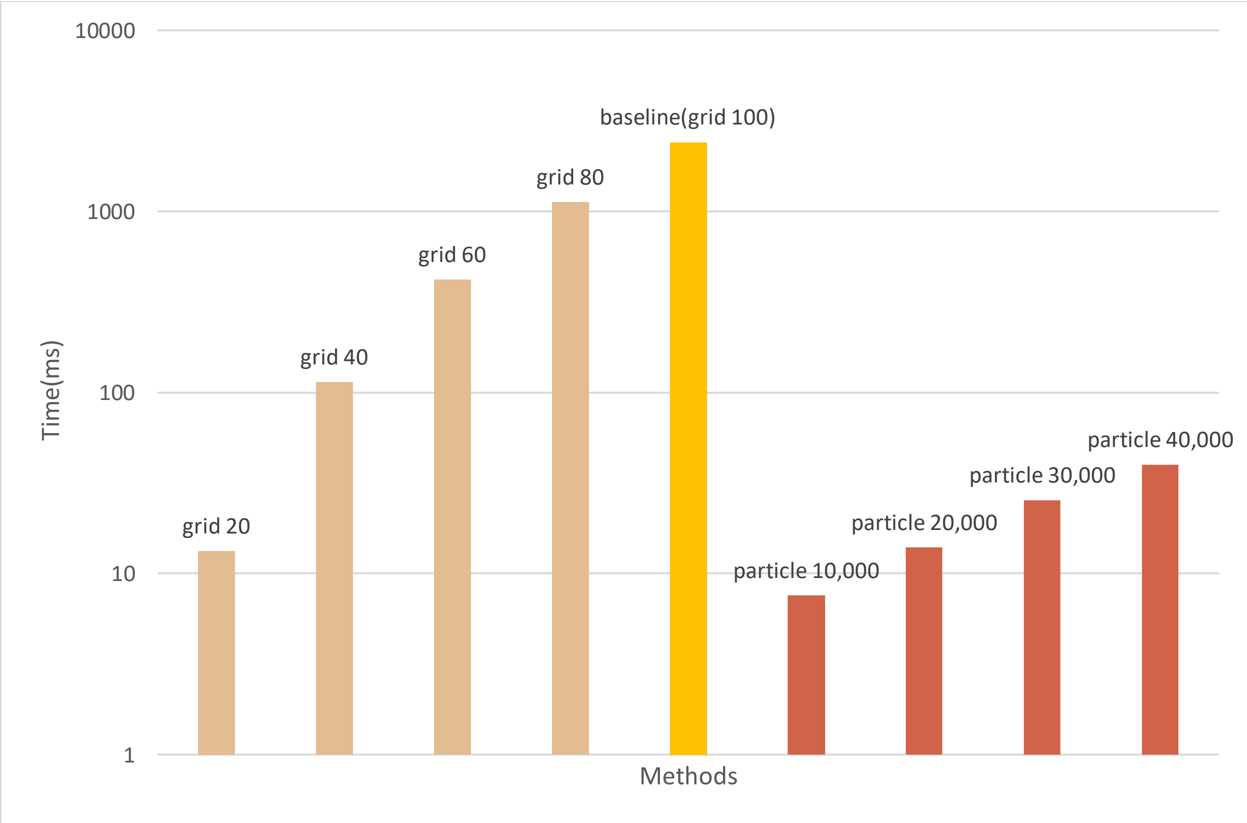

Figure 25. Chart of average execution time of different sets of simulations

Combined with the previous figure, we could tell that p30000 and p40000 have significant shorter time than g40 with a similar precision of temperature and velocity field result. This further suggest the potential of particle-based method to do short time airflow simulation instead of grid-based ones.

6 CONCLUSIONS

6.1 Summary of the Research

This study conducts a comparison of different fluid simulation algorithms in both grid-based methods and particle-based ones. After a review of the literature on different methods of fluid simulation like finite volume method (FVM) and smoothed-particle hydrodynamics (SPH), the detailed methodology of this study is introduced. This study's deliverable is a comparison between particle-based and grid-based methods in indoor airflow simulations with HVAC system.

First the project defines 3 different test cases for room air simulation with simplified HVAC system. For each test case, the project uses OpenFOAM to simulate a grid-based method, buoyantSimpleFoam. The block for the room is evenly divided into different. Then the project uses ParaView to visualize the temperature field result and extracts the data for further analysis. For particle-based method, the project uses OpenGL to simulate SPH method with different particle sizes. The temperature field result is visualized directly on the screen using drawcalls. Finally, the visualized results are compared, and the relevant precision is calculated and analyzed.

6.2 Future Work

The particle-based method is conducted on CPU to get reasonable comparison with gird-based method in OpenFOAM, which is also implemented on CPU. The acceleration on GPU is a trend nowadays, so doing a GPU version of SPH airflow simulation and relevant acceleration methods will be an interesting work.

The SPH method implemented in this study is very simple, and the heat transfer equations implemented are not novel methods. With more detailed implementation and novel heat transfer and buoyant simulation, the result of particle-based method will be much better.

The time step of SPH should not be too large due to its nature, so the simulation could only be conducted in a relatively short period. How to make the legal time step larger is also an valuable discussion.

REFERENCES

AIAA (1998). *Guide for the verification and validation of computational fluid dynamics simulations* (AIAA G-077-1998(2002)). American Institute of Aeronautics and Astronautics. Retrieved Aug 5, 2020 from https://doi.org/10.2514/4.472855.001

Barazzetti, L. (2018). Point cloud occlusion recovery with shallow feedforward neural networks, *Advanced Engineering Informatics*, *38*(1), 605-619. Retrieved Jul 20, 2020 from https://doi.org/10.1016/j.aei.2018.09.007

Becker, M., & Teschner, M. (2007). Weakly compressible SPH for free surface flows. In C. Theobalt, C. Rossl, E. Aguiar, & H. Seidel, *Proceedings of the 2007 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, 209-217. San Diego, CA, USA, Aug 3-4, 2007. Goslar, DEU: Eurographics Association. Retrieved July 11, 2020 from https://dl.acm.org/doi/10.5555/1272690.1272719

Calautit, J., K., & Hughes, B., R. (2014). Measurement and prediction of the indoor airflow in a room ventilated with a commercial wind tower, *Energy and Buildings*, *84*(1), 367-377. Retrieved Aug 5, 2020 from https://doi.org/10.1016/j.enbuild.2014.08.015

Cao, S.-J. (2019). Challenges of using CFD simulation for the design and online control of ventilation systems. *Indoor and Built Environment*, *28*(1), 3–6. Retrieved Aug 5, 2020 from https://doi.org/10.1177/1420326X18810568

Chen, J. K., Beraun, J. E., Carney, T. C. (1999) A corrective smoothed particle method for boundary value problems in heat conduction. *Int J Numer Methods Eng*, 46:231–252.

Chen, Q., & Srebric, J. (2002). A procedure for verification, validation, and reporting of indoor environment CFD analyses, *HVAC&R Research*, *8*(2), 201-216. Retrieved Aug 5, 2020 from https://doi.org/10.1080/10789669.2002.10391437

Cleary, P. W. (1998). Modelling confined multi-material heat and mass flows using SPH. *Applied Mathematical Modelling*, *22*(12), 981-993. Retrieved Sep 11, 2020 from http://dx.doi.org/10.1016/S0307-904X(98)10031-8

Li, Q., Yang, S., & Xu, W. (2010, September 22). Natural indoor ventilation tester of building model. China National Intellectual Property Administration CN2009202958698U

McLean, D. (2013). *Understanding Aerodynamics, Arguing from the Real Physics* (1st ed.). John Wiley & Sons Ltd. Retrieved Aug 5, 2020 from https://play.google.com/books/reader?id=_DJuEgpmdr8C&printsec=frontcover&source=gbs_vpt_reviews&pg=GBS.PA186

Endalew, M., A., Hertog, M., Delele, M., A., Baetens, K., Persoons, T., Baelmans, M., Ramon, H., Nicolai, B., M., & Verboven, P. (2009). CFD modelling and wind tunnel validation of airflow through plant canopies using 3D canopy architecture, *International Journal of Heat and Fluid Flow*, *30*(2), 356-368. Retrieved Aug 5, 2020 from https://doi.org/10.1016/j.ijheatfluidflow.2008.12.007

Harada, T., Koshizuka, S., & Kawaguchi, Y. (2007). Smoothed particle hydrodynamics in complex shapes. In M. Sbert, & S. Spencer (Ed.), *SCCG '07: Proceedings of the 23rd Spring Conference on Computer Graphics*, 191-197. Budmerice, Slovakia, April 26-28, 2007. New York, NY, USA: Association for Computing Machinery. Retrieved Jun 20, 2020 from https://doi.org/10.1145/2614348.2614375

Jones, P., J., & Whittle, G., E. (1992), Computational fluid dynamics for building air flow prediction - current status and capabilities, *Building and Environment*, *27*(3), 321-38. Retrieved July 11, 2020 from https://doi.org/10.1016/0360-1323(92)90033-L

Macklin, M., & Muller, M. (2013). Position based fluids. *ACM Transactions on Graphics*, *32*(4), 104:1-104:12. Retrieved July 11, 2020 from https://doi.org/10.1145/2461912.2461984

Macklin, M., Muller, M., Chentanez, N., & Kim, T. (2014). Unified particle physics for real-time applications. *ACM Transactions on Graphics*, *33*(4), 153:1-153:12. Retrieved Jun 20, 2020 from https://doi.org/10.1145/2601097.2601152

Milne-Thomson, L., M. (1973). *Theoretical aerodynamics* (1st ed.). Dover Publications. Retrieved Jul 20, 2020 from https://books.google.com/books?id=NdDdCgAAQBAJ

Monaghan, J., J. (1992). Smoothed particle hydrodynamics. *Annual Review of Astronomy and Astrophysics*, *30*(1), 543-574. Retrieved Jun 20, 2020 from https://doi.org/10.1146/annurev.aa.30.090192.002551

Morrison, I., B. (2000). *The adaptive coupling of heat and air flow modelling within dynamic whole-building simulation*, University of Strathclyde, PhD Thesis, Glasgow UK

Muller, M., Heidelberger, B., Hennix, M., & Ratcliff, J. (2007). Position based dynamics. *Journal of Visual Communication and Image Representation*, *18*(2), 109-118. Retrieved Jun 19, 2020 from https://doi.org/10.1016/j.jvcir.2007.01.005

Nielsen, P., V. (1974). *Flow in air conditioned rooms*, Technical University of Denmark, PhD Thesis, Copenhagen Denmark

Premžoe, S., Tasdizen, T., Bigler, J., Lefohn, A., & Whitaker, R. T. (2003). Particle-based simulation of fluids. *Computer Graphics Forum*, *22*(3), 401-410. Retrieved July 11, 2020 from https://doi.org/10.1111/1467-8659.00687

Raveendran, K., Wojtan, C., & Turk, G. (2011). Hybrid smoothed particle hydrodynamics. In S. Spencer, *SCA '11: Proceedings of the 2011 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, 33-42. Vancouver, B.C., Canada, August 5-7, 2011. New York, NY, USA: Association for Computing Machinery. Retrieved Jun 20, 2020 from https://doi.org/10.1145/2019406.2019411

Rook, R., Yildiz, M., Dost, S. (2007) Modeling transient heat transfer using SPH and implicit time integration. *Numer Heat Transfer B, Fundam*, *51*(1), 1-23.

Shadloo, M., S., Zainali, A., Yildiz, M., & Suleman, A. (2011). A robust weakly compressible SPH method and its comparison with an incompressible SPH. *International Journal for Numerical Methods in Engineering*, *89*(8), 939-956. Retrieved Jun 20, 2020 from https://doi.org/10.1002/nme.3267

Shao, X., Liao, E., & Zhang, F. (2017). Improving SPH fluid simulation using position based dynamics. *IEEE Access*, *5*(1), 13901-13908. Retrieved Jun 21, 2020 from https://doi.org/10.1109/ACCESS.2017.2729601

Solenthaler, B., & Pajarola, R. (2009). Predictive-corrective incompressible SPH. *ACM Transactions on Graphics*, *28*(3), 40:1-40:6. Retrieved July 11, 2020 from https://doi.org/10.1145/1576246.1531346

Stamou, A., & Katsiris, I. (2005). Verification of a CFD model for indoor airflow and heat transfer, *Building and Environment*, *41*(9), 1171-1181. Retrieved Aug 5, 2020 from https://doi.org/10.1016/j.buildenv.2005.06.029

Wang, L., L., Dols, W., S., & Chen, Q. (2010). Using CFD capabilities of CONTAM 3.0 for simulating airflow and contaminant transport in and around buildings, *HVAC&R Research*, *16*(6), 749-763. Retrieved Aug 5, 2020 from https://doi.org/10.1080/10789669.2010.10390932

Whittle, G., E. (1986). Computation of air movement and convective heat transfer within buildings. *International Journal of Ambient Energy*, *7*(3), 151-164. Retrieved July 20, 2020 from https://doi.org/10.1080/01430750.1986.9675494

Zhai, Z., Chen, Q., Haves, P., and Klems, J., H. (2002). On approaches to couple energy simulation and computational fluid dynamics programs. *Building and Environment*, *37*(8), 857-864. Retrieved July 20, 2020 from https://doi.org/10.1016/S0360-1323(02)00054-9

Zhai, J., Z., Chen, Q., Y. (2004). Performance of coupled building energy and CFD simulations. *Energy and Buildings*, *37*(4), 333-344. Retrieved July 20, 2020 from https://doi.org/10.1016/j.enbuild.2004.07.001

A.1 BLOCKMESHDICT EXAMPLE FOR CASE 1

```
FoamFile
{
   version     2.0;
   format      ascii;
   class       dictionary;
   object      blockMeshDict;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

scale   0.25;

vertices
(
   (0 0 0)
   (1 0 0)
   (2 0 0)
   (3 0 0)
   (4 0 0)

   …

   (0 4 4)
   (1 4 4)
   (2 4 4)
   (3 4 4)
   (4 4 4)
);

edges
(
);

blocks
(
   hex (0 1 6 5 25 26 31 30) (10 10 40) simpleGrading (1 1 1)
   hex (1 2 7 6 26 27 32 31) (10 10 40) simpleGrading (1 1 1)
   hex (2 3 8 7 27 28 33 32) (10 10 40) simpleGrading (1 1 1)
   hex (3 4 9 8 28 29 34 33) (10 10 40) simpleGrading (1 1 1)

…

   hex (15 16 21 20 40 41 46 45) (10 10 40) simpleGrading (1 1 1)
```

```
    hex (16 17 22 21 41 42 47 46) (10 10 40) simpleGrading (1 1 1)
    hex (17 18 23 22 42 43 48 47) (10 10 40) simpleGrading (1 1 1)
    hex (18 19 24 23 43 44 49 48) (10 10 40) simpleGrading (1 1 1)
);

boundary
(
    frontAndBack
    {
        type wall;
        faces
        (
            (20 21 46 45)
            (21 22 47 46)
            (22 23 48 47)
            (23 24 49 48)

…

            (0 5 30 25)
            (5 10 35 30)
            (10 15 40 35)
            (15 20 45 40)
        );
    }

    topAndBottom
    {
        type wall;
        faces
        (

    (0 1 6 5)
    (25 26 31 30)
    (1 2 7 6)
    (26 27 32 31)
    (2 3 8 7)
    (27 28 33 32)
    (3 4 9 8)
    (28 29 34 33)

…

    (15 16 21 20)
    (40 41 46 45)
    (16 17 22 21)
```

```
   (41 42 47 46)
   (17 18 23 22)
   (42 43 48 47)
   (18 19 24 23)
   (43 44 49 48)

      );
   }

   hot
   {
      type patch;
      faces
      (
         ( 31 32 37 36)
      );
   }
   cold
   {
      type patch;
      faces
      (
         (9 14 39 34)
         (14 19 44 39)
      );
   }
);
mergePatchPairs
(
);
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //
```

A.2 BOUNDARY CONDITION SETTING FILE EXAMPLE FOR TEMPERATURE

```
FoamFile
{
   version     2.0;
   format      ascii;
   class       volScalarField;
   location    "0";
   object      T;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

dimensions      [0 0 0 1 0 0 0];

internalField   uniform 293;

boundaryField
{
   frontAndBack
   {
      type           zeroGradient;
   }

   topAndBottom
   {
      type           zeroGradient;
   }

   hot
   {
      type           fixedValue;
      value          uniform 307.75; // 34.6 degC
   }

   cold
   {
         type              inletOutlet;
         inletValue        uniform 307.75;
         value             uniform 293;
   }
}

// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //
```

A.3 SIMULATION SETTINGS IN CONTROLDICT FILE

```
FoamFile
{
   version     2.0;
   format      ascii;
   class       dictionary;
   object      controlDict;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

application     buoyantSimpleFoam;
startFrom       startTime;
startTime       0;
stopAt          endTime;
endTime         10;
deltaT          0.01;
writeControl    timeStep;
writeInterval   50;
purgeWrite      20;
writeFormat     ascii;
writePrecision  6;
writeCompression off;
timeFormat      general;
timePrecision   6;
runTimeModifiable true;

// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //
```

A.4 VISUALIZED RESULT OF TEMPERATURE IN TEST CASE 2 AND 3

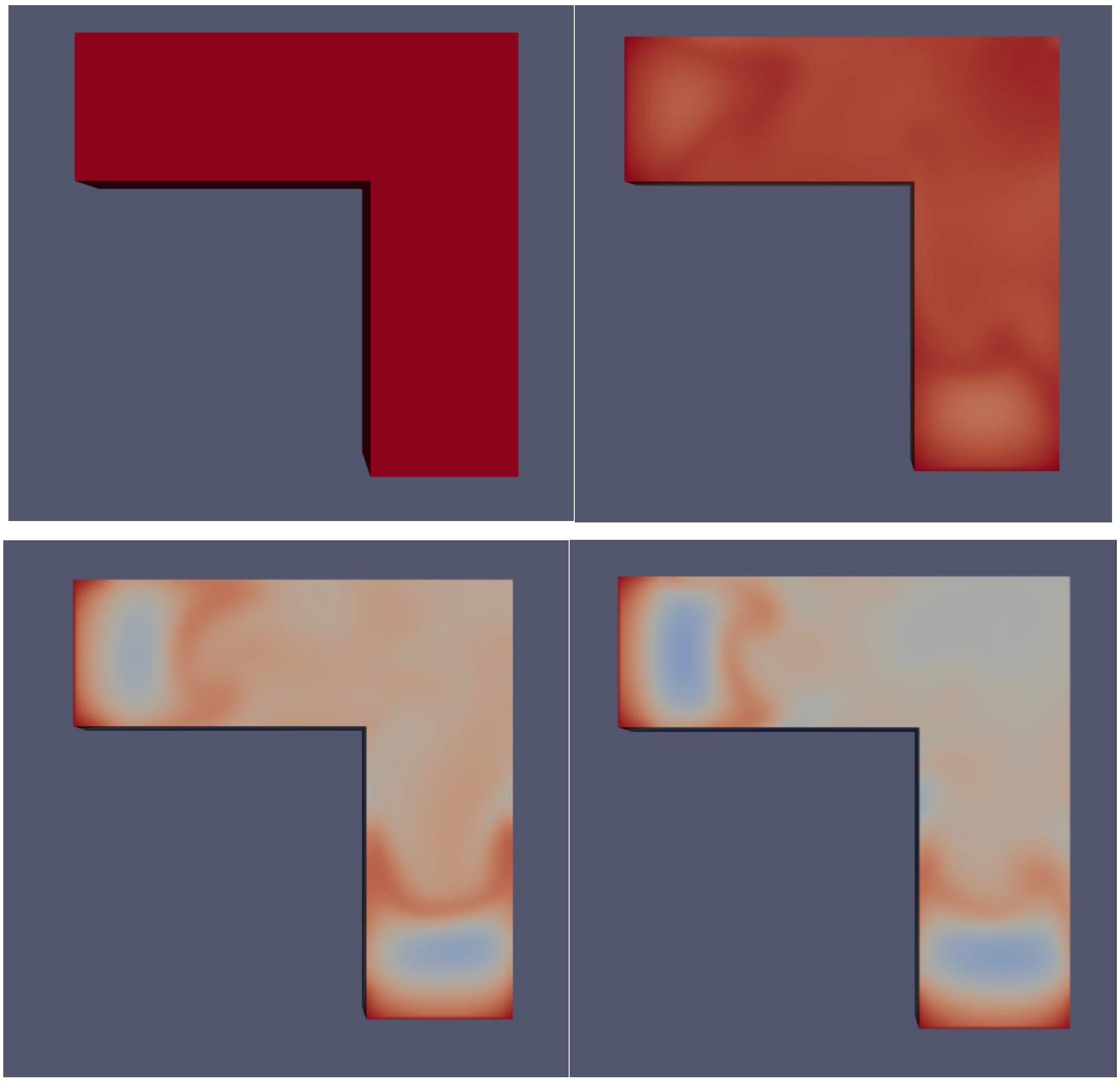

Figure 26. Temperature visualization of test case 2 with cell size of 20(top left), 40(top right), 60(bottom left), 80(bottom right)

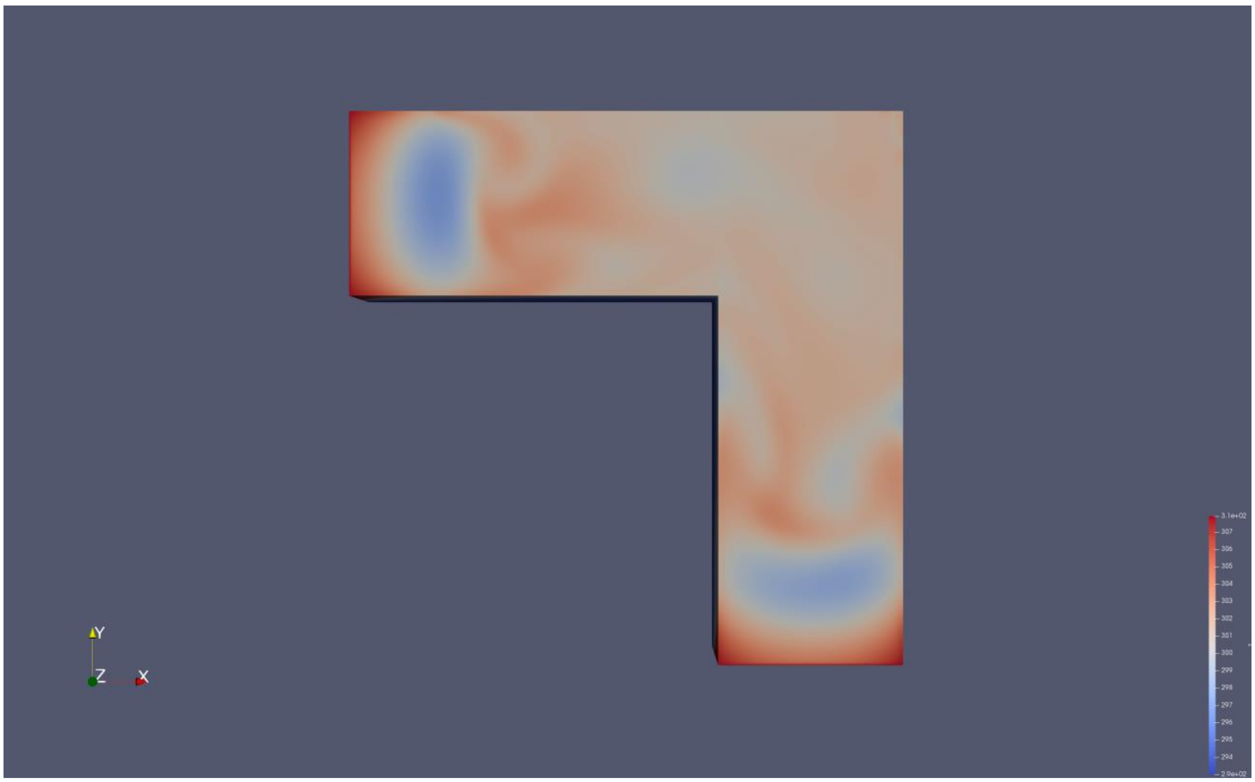

Figure 27. Temperature visualization of test case 2 with cell size of 100

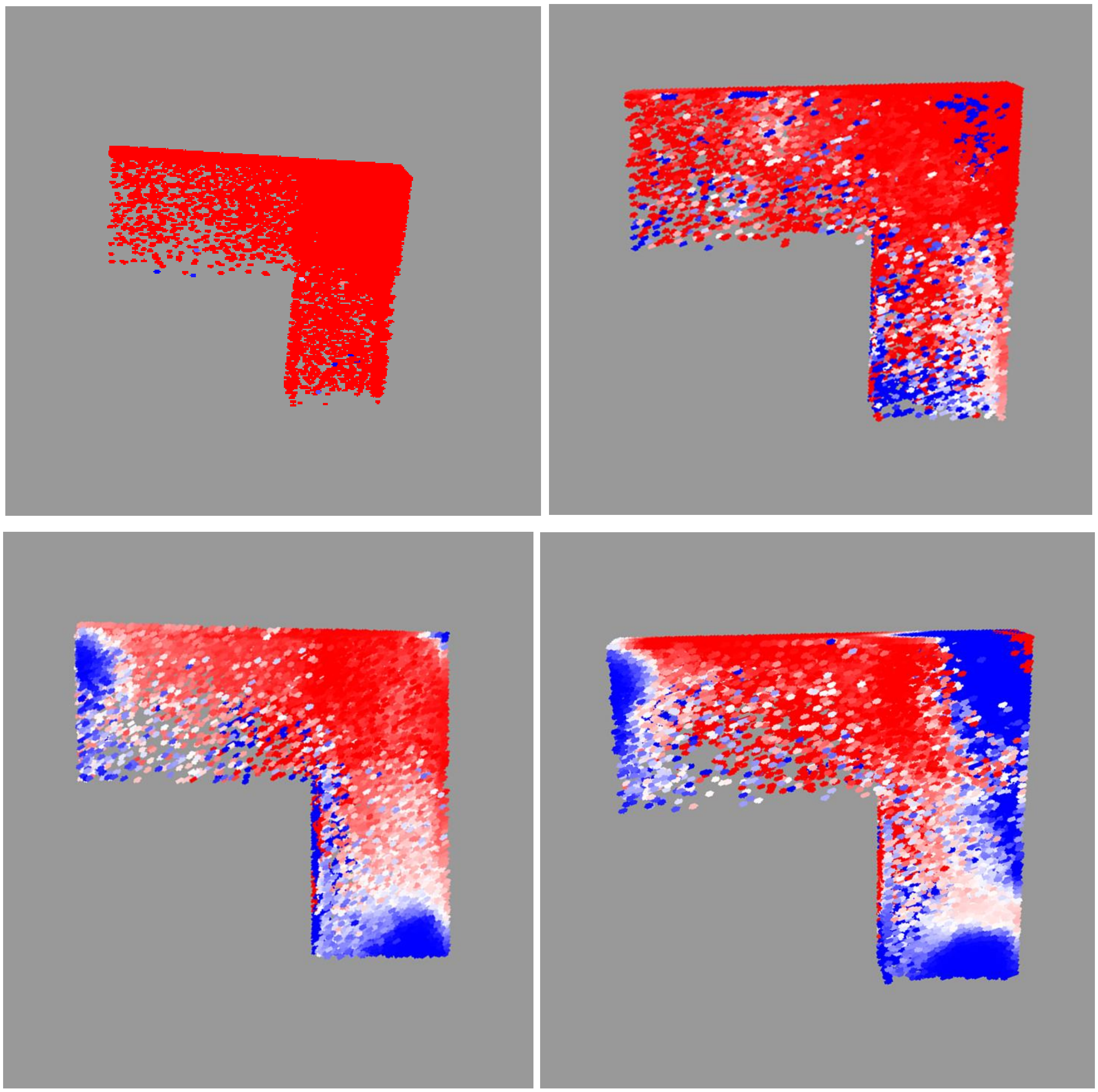

Figure 28. Temperature visualization in SPH of test case 2 with particle number of 10,000(top left), 20,000(top right), 30,000(bottom left), 40,000(bottom right)

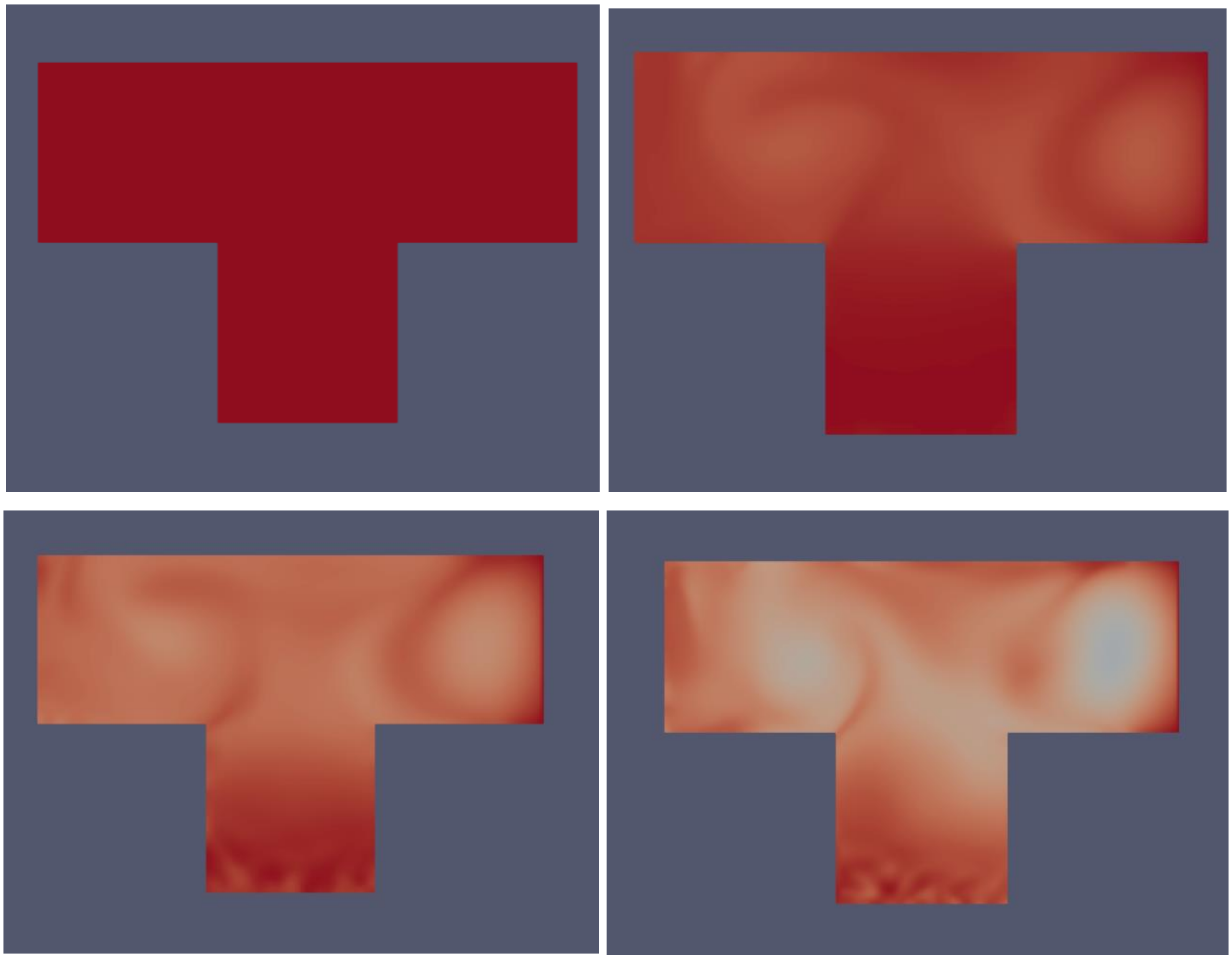

Figure 29. Temperature visualization of test case 3 with cell size of 20(top left), 40(top right), 60(bottom left), 80(bottom right)

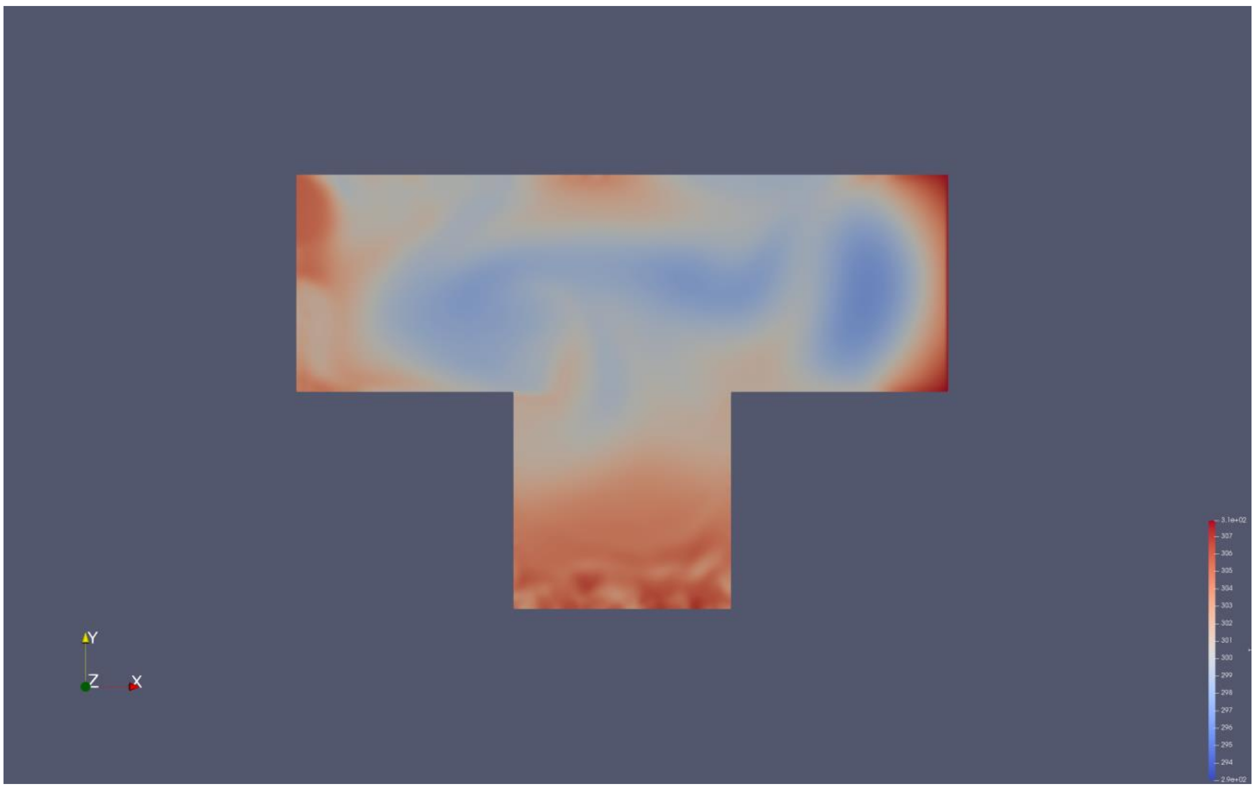

Figure 30. Temperature visualization of test case 3 with cell size of 100

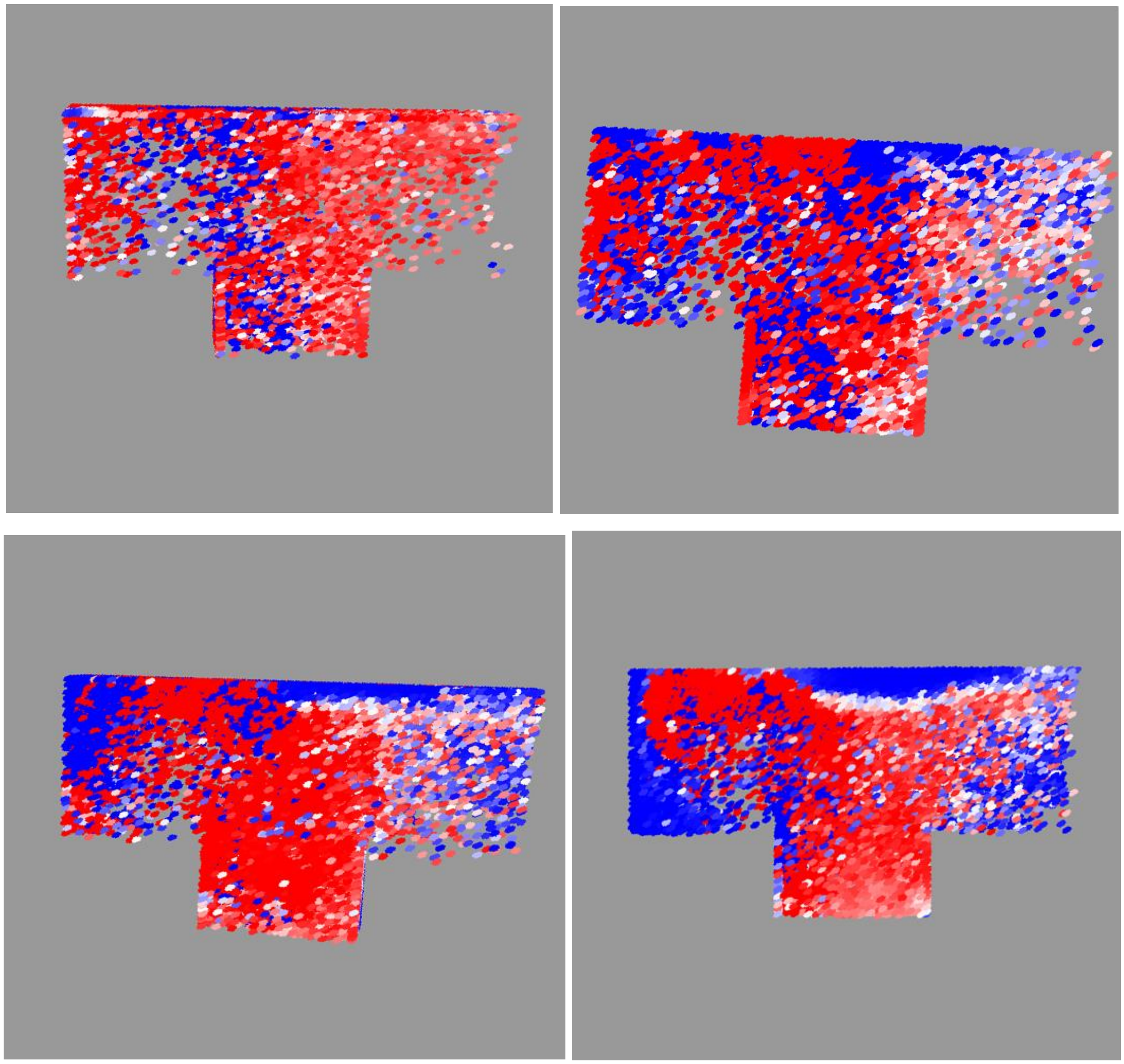

Figure 31. Temperature visualization in SPH of test case 3 with particle number of 10,000(top left), 20,000(top right), 30,000(bottom left), 40,000(bottom right)

A.5 AVERAGE EXECUTION TIME FOR DIFFERENT SETS OF SIMULATIONS

Table 3. Execution time for simulations

TestCase	Method	GridSize/particleNumber	ExecutionTime
Block	SbuoyantSimpleFoam	20	12.03
		40	109.88
		60	415.23
		80	1117.44
		100	2495.7
	SPH	10000	8.06
		20000	12.47
		30000	26.48
		40000	38.39
L-Room	SbuoyantSimpleFoam	20	13.32
		40	114.05
		60	419.82
		80	1122.9
		100	2405.29
	SPH	10000	7.54
		20000	13.88
		30000	25.46
		40000	40.02
T-Room	SbuoyantSimpleFoam	20	12.77
		40	106.29
		60	403.35
		80	1149.58
		100	2340.85
	SPH	10000	10.234
		20000	15.84
		30000	28.57
		40000	45.68