Team Control Number

22875633

Problem Chosen

D

2022

IMMC

Summary Sheet

In this paper, we developed a system that can be used to build a virtual Hong Kong in the virtual world. To ensure that the system is precise, coordinates were to be determined first, and due to the size of the area, we chose the Hong Kong 1980 coordinates, which is also used to draw civilian maps in Hong Kong. The height is determined by the Standard Sea Level. Thus, the position of an element can be determined in Hong Kong.

After that, we developed the basic elements of the system. Similar to the game *Minecraft*, elements are divided into two parts: shapes and entities. Shapes are used to determine objects that do not move a lot, like buildings and terrain, and entities for the rest, for example, transportation and animals. We developed some properties to describe them. There are certain relationships between them, and some of these are considered as properties as well.

To determine the calculation need for a "refresh", we first need to determine what is a "fit" for a certain set of elements. Q-Learning algorithm and K-means algorithm are perfect for these conditions. Hence, the calculation of the system is easy to be determined.

We divided a "refresh’ " into two parts, "shapes" refresh, which is to be done under the instruction of humans, and "entity" refresh, which are done automatically. The pixels and distance between the observer and the object, and the movement of the object are considered, and some indexes describing lag and clearance are defined. Therefore, the calculation amount can be simplified into a simple equation, and data were obtained via the Internet and Satellite Images. Finally, we get the results to the questions proposed in the problem.

Building a virtual Hong Kong in metaverse

Team#22875633

January 24, 2022

Contents

1 Introduction

1.1 Background

Nowadays, *metaverse* has become a heated topic and a symbol of development in virtual technology. It refers to a three-dimensional virtual world, especially in online role-playing games such as *Second Life*, *Minecraft* and *Roblox*. In this system, people can build virtual items according to their own needs and complete basic communication with others such as buying and selling items from others as well as sharing information of their creations. In other words, this is a society which only depends on the instructions of game players. Using this technology, we can simulate the society by uploading the data given and create the items which exist in daily life to predict the future development of this system.

In the three games mentioned above, players use limited basic blocks and their own imagination to build their own virtual world. It can be seen that the type of basic blocks is very important. They must meet the needs of construction to the greatest extent, and at the same time, the number of types and size should be appropriate so that players can easily understand the function of each blocks and make full use of them.

Now Hong Kong is also confronted with the problem of how to improve its urban construction without imposing policies on the real city. On this occasion, creating a virtual Hong Kong by using metaverse is key to this challenge. We can create simple elements in this virtual world which fit the reality so that government can observe the effect of their policies by giving basic instructions in this virtual city. In this way, there will be no loss in the real world due to strategic mistakes because all of the mistakes actually happen in real life. City construction in Hong Kong can be easier and more efficient.

1.2 Problem Restatement

To build a virtual city and make sure that it can be used in daily thing, we need to figure out its basic elements and find out the computing and storage resources needed to maintain a virtual Hong Kong. So our work is divided into 2 parts:

- Build the basic blocks that satisfy the needs of citizens and can be seen in a real city. We will explain the functions of each block and how it can be planted in the virtual city.
- Calculate the fitting degree of basic blocks to make sure our virtual city can simulate Hong Kong.
- Describe the computing and storage resources required in our design by calculating the speed at which it update the state of moving objects (such as pedestrians and cars) and how much bytes of storage it needs to maintain a virtual city.

1.3 General Assumptions

- **The ground of Hong Kong can be regarded as a plain.**

  Though we all know the earth is an ellipsoid, Hong Kong is so small that we can regard its ground as a plain. Also, the buildings in Hong Kong are generally not high. So we can neglect factors of ground inclination.

- **The data will be updated non automatically when serious natural disasters such as typhoons and hurricanes destroy buildings.**

  We divide the shifts in location and shape of blocks into two types: small-scale shifts such as car driving and large-scale shifts such as decay of buildings. The latter happen rarely, so the data can be uploaded non automatically.

2 Model A

2.1 Model Overview

In this model, we will mainly discuss the basic block sets in this society. We divide the process into two parts: how to describe the location and features of the blocks and how to make sure our blocks fit the constructions in real Hong Kong. we will find that what we need to build is divided into two parts: blocks with static attributes and blocks mobile entities which we call *entities*. Later we will discuss the relationship between them.

When we take everyday life into consideration, we find the common blocks in a city. The first two types are entities and the last one belongs to static attributes.

- **Transportation vehicles**
  In Hong Kong, there are four types of transport: rail transit, air transport, transport on sea and road transport.

- **Pedestrians**
  It's common to see pedestrians walking in Hong Kong.

- **Buildings**
  These buildings consists of building blocks and are seen as static objects.

In this way, we can build a basic structure of our virtual city.

2.2 Notation

f(block)	properties of the block
V	volume of the block
ρ_1	continuity of lines in patterns
ρ_2	density of lines in patterns
L	dimension of lines in patterns
α_{radius}	a certain proportion which determines L
r	radius of the dots in patterns
θ_1	angle with O_x
θ_2	angle with O_y
k	a special property of each block
a	common parameter of the geometry
b	common parameter of the geometry
h	common parameter of the geometry
α	common parameter of the geometry
β	common parameter of the geometry
(x,y,z)	the position in the three-dimensional coordinate

2.3 Creating the Blocks

In our model, we use a coordinate system to describe the absolute location of each blocks which can enable us to quickly track the route of each moving block. Also, we use a matrix which is made up of the properties: volume, stripes, colors and shape.

We use a shape set to describe the geometric features of blocks. There are several kinds of geometrical configurations.

Type 0 is hexahedron, which is shown in the picture below. α, β, γ, a, b, c, θ_1, θ_2 can be introduced to describe its shape. Also,k is introduced to describe its dimension. It means that, when k=0, it's a pyramid while when k=1, it's a parallelepiped.

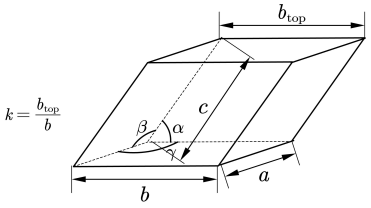


Fig.1 An example of parallelepiped

Type 1 is ellipsoid. a, b, c are introduced to determine its shape.

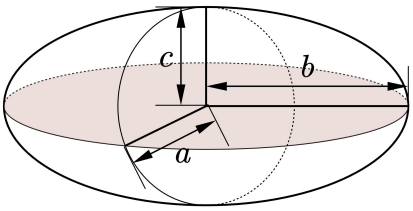


Fig.2 An example of ellipsoid

Type 2 is elliptic column platform. This time, a, b, h, α, β, k is introduced to describe its shape. When $k = 1$, the block is a elliptical italic and when $k = 0$, the block is a elliptical colic.

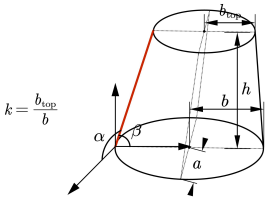


Fig.3 An example of elliptic column platform

Secondly, we need to describe the color and patterns on the blocks. We use three primary colors(red, blue and yellow) to describe the color of blocks. For the patterns of the blocks, we adopt two indicators ρ_1 and ρ_2 to respectively describe the features of these patterns; for the coloring factor, we use the international standard of (R,G,B) color base.

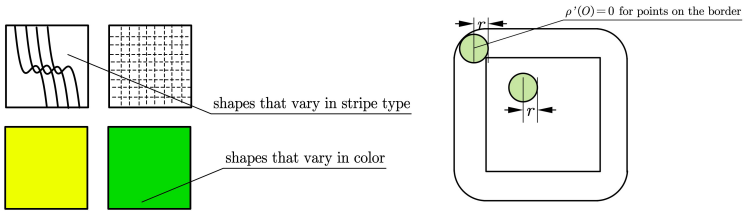


Fig.4&5 Patterns on the blocks

According to the picture, we can know the correct definition of r and L. We use α_{radius} to determine the proportion of this dimension. α_{radius} should be at a moderate size to prohibit regional distortion of the image.

$$r \overset{def}{=\!=} L \times \alpha_{\text{radius}} \tag{2.1}$$

Therefore, the proportion of stripes in the drawn circle can be calculated as:

$$\rho'(o) = \frac{\text{Line Area}}{\pi r^2} \tag{2.2}$$

It's important to note that the hook face of the stripes is continuous. In the picture below, we can see that for parts that intersected with the border, we'll move the parts at a distance of $d \to 0$, the position of the parts do not change. So when we observe a specific part move as in Fig.14, the change in its area can be regarded as a continuous linear change. Therefore, the function is continuous at all points.

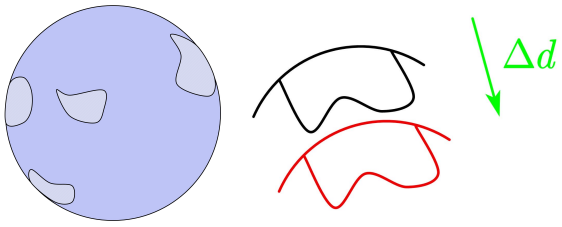


Fig.5&6

Now we can define the two indicators introduced as before. Continuity, which in other words, can be described as measured as following. Whereas density can be approximately seen as the average $\rho'(0)$ after being integrated on a two-dimensional scale.

$$\rho_1 \overset{def}{=\!=} \frac{\max\limits_{\text{Surface}} \rho'(o) - \min\limits_{\text{Surface}} \rho'(o)}{L} \tag{2.3}$$

$$\rho_2 \overset{def}{=\!=} \frac{\iint_{Surface} \rho'(x,y)d\sigma}{S(\text{Surface})} \tag{2.4}$$

Thirdly, we will take its volume into consideration. Its volume is easy to be calculated according to its different geometric features.

So, when we combine all the properties together, we can get the matrix:

$$\text{f(block)} = \begin{pmatrix} \text{V} & & \\ & (\rho_1, \rho_2) & \\ & & (\text{R}, \text{G}, \text{B}) \\ & & \text{vectors in the shape set} \end{pmatrix}$$

(In this part, vectors refer to their determining factors of the block's shape as mentioned at the beginning of section 2.3.)

3 Calculating the Fitting Degree

3.1 Problem Overview

In this part, we will discuss the fitting degree of our blocks. Now that we have established a characteristic indicator (shown in a matrix) to represent each block, we can

now consider the process of fitting them into a complete architectural structure. Suppose there are numerous "blueprints" automatically devised by our algorithm (which will be introduced later), we need to evaluate its "excellence" for a specific building.

Therefore, to quantify this "how good" factor, we set up a "fitting degree", which can be implemented to get the best "blueprint" and do the required calculations. Most of the indicators introduced below are related to f(block) itself, which makes the calculations more accurate.

3.2 Notation

F_1	the fitting degree in volume
F_2	the fitting degree in lines
F_3	the fitting degree in color
F_4	the fitting degree in shape
ϕ	the fitting degree of the whole block
$Overlap_{Cluster}$	the degree in which blocks and items overlap
$Excessive_{Cluster}$	the degree in which blocks vary with items
$\Delta\rho_1$	change in continuity
$\Delta\rho_2$	change in density
Δ_e	the maximum color difference distinguishable by human eyes
Δ_{color}	color difference of blocks
I	definition of shape
Δ_{shape}	shape difference of blocks

3.3 Simulating the Shape

Firstly, we need to calculate the fitting degree of the block's shape. A brick too small or large in size certainly does no good to our construction. Consider the overlapped and the outside parts:$Overlap_{Cluster}$ and $Excessive_{Cluster}$. So we can the fitting degree shape can be described as:

$$F_1 = \frac{Overlap_{Cluster} - Excessive_{Cluster}}{V} \tag{3.1}$$

Secondly, we will discuss the fitting degree in patterns on blocks. It will be divided into lines, colors and shape, which will be easy to calculate with f(block) as introduced before. $\Delta\rho_1$ and $\Delta\rho_2$ are introduced to explain the relationship between blocks and items. So we can get the following formulae:

$$\Delta\rho_1 = |\rho_1(surface(Block)) - \rho_1(surface(Target))| \tag{3.2}$$

$$\Delta\rho_2 = |\rho_2(surface(Block)) - \rho_2(surface(Target))| \tag{3.3}$$

$$F_2 = e^{-(k_1\Delta\rho_1 + k_2\Delta\rho_2)} \tag{3.4}$$

Under the circumstance where k_1 and k_2 are constant weights, the indicator declines at exponential speed as $\Delta\rho_1$ and $\Delta\rho_2$ increases, meaning that there is little tolerance of the disparity in these two factors between the target and the actual block.

When we discuss the fitting degree in color, we identify Δ_e as the maximum color difference distinguishable by human eyes. So when we compare $(R_{\text{Block}}, G_{\text{Block}}, B_{\text{Block}})$ with $\left(R_{\text{Target}}, G_{\text{Target}}, B_{\text{Target}}\right)$, similar to the pattern part, we define:

$$\Delta_{color} \overset{def}{=\!=} \sqrt{\sum_{R,G,B} \frac{\left(R_{Block} - R_{Target}\right)^2}{3}} \tag{3.5}$$

$$F_3 = e^{1-\frac{\Delta_{color}}{\Delta_e}} \tag{3.6}$$

(When $\Delta_{color} = \Delta_e$, the fitting degree is 100%.)

The last factor is shape. We assume that in calculation, the type the block belongs to must be consistent with the building's shape. It means that if the shape of block is consistent with the type in calculation, I=1. Otherwise I=0. So we can find that:

$$\Delta_{\text{shape}} = \sqrt{\frac{\sum_{x \in \text{attribute\_set(Block)}} |x_{\text{Block}} - x_{\text{Target}}|}{\#\text{attribute\_set}}}$$

$$F_4 = I \cdot e^{-\Delta_{\text{Shape}}}$$

$$F_1 = \frac{\text{Overlap (Cluster)} - \text{Excessive (Cluster)}}{V}$$

With all the previous factors combined, we get a "vector" in the same 4×4 "format" as f(block), for easy calculation. Multiply the pre-factor matrix by f(block).

$$\phi = (F_1, F_2, F_3, F_4) \overset{def}{=\!=} (V_{\text{v}}, V_{\text{line}}, V_{\text{c}}, V_{\text{sh}}) \tag{3.7}$$

$$\begin{pmatrix} V_v & & & \\ & V_{st} & & \\ & & V_c & \\ & & & V_{sh} \end{pmatrix} \times f(\text{Block}) = \begin{pmatrix} V_v & & & \\ & V_{st} & & \\ & & V_c & \\ & & & V_{sh} \end{pmatrix} \times \begin{pmatrix} V \overset{\text{def}}{=\!=} J_1 & & & \\ & (\rho_1, \rho_2) \overset{\text{def}}{=\!=} J_2 & & \\ & & (R,G,B) \overset{\text{def}}{=\!=} J_3 & \\ & & & (\alpha_1, \ldots, \alpha_k) \overset{\text{def}}{=\!=} J_4 \end{pmatrix}$$

$$= \begin{pmatrix} V_v J_1 & & & \\ & V_{st} J_2 & & \\ & & V_c J_3 & \\ & & & V_{sh} J_4 \end{pmatrix} = \begin{pmatrix} V_{vV} & & & \\ & (V_{st}\rho_1, V_{st}\rho_2) & & \\ & & (V_c R, V_c G, V_c B) & \\ & & & (V_{sh}\alpha_1, \ldots, V_{sh}\alpha_k) \end{pmatrix}$$

$$\overset{\text{def}}{=\!=} f_{\text{same}}(\text{Block}, \text{Cluster})$$

The result of the calculation is a sort of "compromise"————the balance point of the current set of blocks and the actual building. To illuminate this concept, we introduce the

definition of the "same" blocks of a building don't necessarily exist, they can be used to equally quantify the consequent calculations.

Finally, we can get the fitting degree:

$$\alpha_{\text{fit}} = \frac{\sum_{Block} \frac{f_{\text{same}}(Block, Cluster)}{f_{\text{same}}(Block, Building)}}{\text{Card(blocks)}} (0\%, 100\%) \tag{3.8}$$

3.4 Combining the Degree of Each Block

After we come up with the 'same block' notion, we can actually calculate how much kinds of blocks it takes to construct the virtual world. Firstly, we need to try this on a smaller scale. Consider a single building. We've already got the 'same block' of each block of the building. However, the disparities of some blocks may only vary so little on the numerical scale that the difference is not detectable by the human eye. We define the difference between 'same block's as

$$\Delta_{\text{same}}(Block1, Block2, Cluster) = \sqrt{\frac{\Sigma\,(\text{difference of four factors})^2}{4}} \tag{3.9}$$

and suppose the least detectable value of this by naked eye equals δ_{eye}, which can be calculated with enough information or experiments.

Algorithm 1 Cluster Algorithm

```
for part number j of the total m parts: do
    parameters ← the attributes of part j
    if the distance between the parameters vector and the average parameters vector of
    geometric form i: then
        form i ← part j
        average parameters of form i ← new average considering part j
    else
        new form ← part j
    end if
end for
show the total amount of geometric forms
```

Now that we've figured out all the features of basic blocks and how to calculate its fitting degree, we need to calculate the running speed of our virtual society so that it can be put into real calculation. In this model, we fill focus on the byte required in different moving circumstances and different transportation devices. In addition, we will calculate its lagging index according to its updating speed.

Now all we need to do is to calculate the ideal number of blocks. Therefore, we use the K-means algorithm, where a cluster with a 'distance' of points less than δ_{eye} can be

aggregated as one block. Running the following algorithm can help us get the number of blocks of a specific building. Repeat this on all buildings, and again use this algorithm, deriving the 'same same' blocks, the amount of which is what we want.

From section 3.3 we get the fitting degree of a specific building plan. So next we need an algorithm to search the optimal one among the plans. In order to achieve this without repeated or excessive calculations, we use Q-Learning technology to help us realize the whole process. This can make calculations in the second model easier. The steps are shown below.

Algorithm 2 Q-Learning algorithm

```
divide the whole area into several cubic parts
for part (i, j, k): # simulate the single part do
    initialize Q table # the potential benefits of the plan, a matrix
    initialize S # the set of actions to simulate the city with geometric forms, a list
    initialize  and  #  is the rate of learning while  is the rate of declining. Both  and  are
    in the range of (0, 1).
    choose action A from S due to a policy from Q # A is a plan of fitting the blocks with
    geometric forms, choose the plan with the highest simulation level
    while not done: do
        take action A, observe R, S'
        Q(S,A)←Q(S,A)+[R+maxa Q(S ',a)Q(S,A)]# update the Q table and the status S
        ←S'
    end while
end for
for the borders of parts: do
    small adjustions to simulate the area better
end for
```

4 Model B

4.1 Model Overview

For model 2, we need to give a rough estimate of the performance of the virtual city. We divide this into that of the cloud, i.e, global capability and that from a user's perspective. Meanwhile, rough calculations and several indicators will be set up to aid our estimate.

4.2 Assumptions

- **All the straps and lines have width.**
- **All the entities can be seen as a particle.**

To fasten the calculating speed, we assume that all the entities' volume can be neglected. In other words, thy can be seen as particles.

- **There is a "middle" area between blocks and entities.**

  There are places that people can enter but cars cannot and certain areas that only special transportation vehicles can enter. To make it easier for calculation, a "middle" area is set between blocks and entities.

- **If the distance from one entity to a block is constant from any perspective, the measure of the area it has observed is constant.**

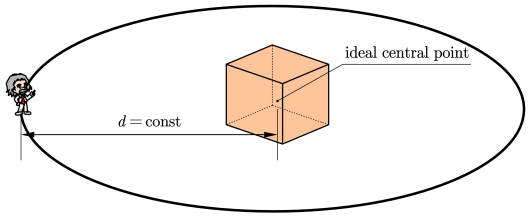


Fig.6 Explanation of constant measure

- **Entities don't keep cars out.**

  After observing the traffic condition in Hong Kong, we find out that there aren't much cars in Hong Kong. So we assume that the entities will not affect cars.

- **Only flights recorded in HKG are considered in this model.**

  We find that most of the fights passing Hong Kong arrive or take of at HKG while there are no planes observed on Hong Kong's territory. Therefore, we assume that only flights recorded in HKG has an impact.

- **All the data are collected after the pandemic.**

  To make sure that our calculation matches the latest information, all the data will be collected after the pandemic (2020).

- **Observation of planes can be identified as a two dimensional picture.**

  Considering the assumptions above, there will be no planes flying over the HKG in this model, which means if a person want to observe a plane, its flying route must be flush with line of sight.

- **We take the standard horizontal plane as the datum.**

4.3 Obtaining Data

As we need to build the virtual Hong Kong as real as possible, we need some data provided by the government to make the virtual Hong Kong look like Hong Kong in real life.

Firstly, we need the information of the buildings in Hong Kong, as we need to place the blocks in the virtual Hong Kong. Meanwhile, we need the data about Hong Kong' s population and transportation to estimate how much calculation the virtual Hong Kong needs. This kind of data include data of Hong Kong' s population, total amount of cars, total traffic volume and total area. These data can be gotten from the government. In addition, we need the data of human eye resolution to confirm an adequate resolution for the virtual Hong Kong. Fortunately, he data can be found from the Internet.

4.4 Notation

α_{lagging}	lagging index
α_{c}	clarity index
P_{reload}	pixels needed for reload
P_{target}	pixels targeted for use
$P_{\text{calc\_cloud}}$	the number of calculations needed from the cloud
$P_{\text{calc\_local}}$	the number of calculations needed locally
K	a specific index to evaluate storage leve; of this virtual city
x	a variable related to reloading speed
y	a variable related to clarity degree
V_{reload}	reloading speed
V_{eye}	eye's reacting speed
C	clarity degree of blocks
C_{eye}	clarity degree of human eye
V_{E}	normal moving speed of an entity
L_{E}	dimension of an entity
S_{E}	measure of an entity
d	distance to the observer
$\Delta_{d_{\text{E}}}$	the distance an entity covers during one reload
$\Delta_{V_{\text{E}}}$	the changing volume of an entity during one reload
N_{E}	number of pixels an entity require
ρ_{E}	density per unit area of an entity
δ_{E}	proportion of entities in observable area
r_{g}	observation range of man's eye
r_{street}	observation range of man's eye on streets

4.5 Calculating Local Speed

We need to calculate the reloading speed (V_{reload}). In this model, a reload refers to one refreshing "actions" performed by the computer, the amount of which per second (defined in the essay as V_{reload}) can directly affect a user's experience. Therefore, we use a "lag" index to show how much a computer lags behind human eyes. As there is little tolerance of the index, we again use exponential decline where the indicator amounts to zero when the reloading speed of the computer is conspicuously detectable by the human eyes. The process is divided into several parts.

First, we need to calculate the lagging index. The formula is shown below. (In this formula,V_{eye} refers to human eyes' reacting speed,which is about 16 Hz.)

$$x = \frac{V_{\text{reload}}}{V_{\text{eye}}} \tag{4.1}$$

$$\alpha_{\text{lagging}} = \begin{cases} 1 & , x \in (0, \infty) \\ 2 - e^{1-x} & , x \in \left[\ln \frac{e}{2}, 1\right] \\ 0 & x \in \left(0, \ln \frac{e}{2}\right) \end{cases} \tag{4.2}$$

Secondly, we need to calculate the clarity index. "Clarity" is defined as the number of pixels exposed to a user when he or she watches an area of 1 square meter 1m ahead of him or her. The scale "1m" follows that in the game. In this part, we identify$C$ as the clarity degree now and C_{eye} as the clarity degree human eyes can distinguish. According to research data, the clarity degree human eyes can distinguish is about 40960000. The index can be calculated by the formula below:

$$y = \frac{C}{C_{\text{eye}}} \tag{4.3}$$

$$\alpha_{\text{c}} = \begin{cases} 1 & , y \in (1, \infty) \\ -\left(\frac{4096}{4095}\right)^2 (y-1)^2 = 1 & , y \in \left[\frac{1}{4096}, 1\right] \\ 0 & , y \in \left(0, \frac{1}{4096}\right) \end{cases} \tag{4.4}$$

(The bigger α_{clarity} is, the more comfortable people feel when looking at the virtual city.)

This time, according to visual physics, there is a clear relation between the number of pixels and the square of the distance (which will be discussed later). Therefore, we also use a square indicator in the function, where the index amounts to zero when a 1cm$\hat{2}$ 1m ahead of the user becomes an angle pixel.

In addition, we need to find out the pixels required when a block moves when it's uploaded. First we calculate how much pixels will be uploaded when the city is reloaded

(for example, the observer walks a step or the entity covers an observable amount of distance).

4.5.1 Observing From the Ground

When man observe from the ground, the range that man can observe is about 2 kilometers. As shown in Fig.8&9 and according to basic geometric relations, we can get that:

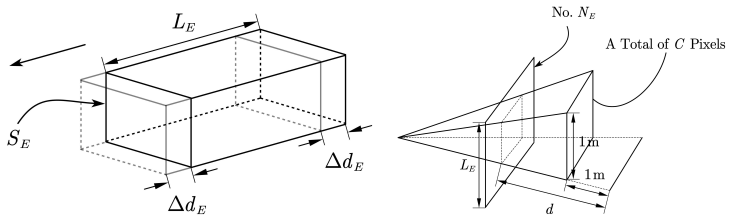


Fig.8&9 State of observer when he moves

$$\Delta_{V_\mathrm{E}} = S_\mathrm{E} \times \Delta_{d_\mathrm{E}} \tag{4.5}$$

$$N_\mathrm{E} = \left(\frac{r-d}{r-1}\right)^2 C \tag{4.6}$$

$$\frac{2\Delta_{d_\mathrm{E}}}{L_\mathrm{E}} = 2\Delta_{d_\mathrm{E}} L_\mathrm{E} C \left(\frac{r-d}{r-1}\right)^2 \tag{4.7}$$

According to the definition of ρ_E there are $N_\mathrm{E} = \pi {r_\mathrm{g}}^2 \rho_\mathrm{E}$ entities in this area. So in an area of which the radius is d,

$$N_\mathrm{g}(d) = \pi d^2 \rho_\mathrm{E} \tag{4.8}$$

So the proportion of the amount of entities within a d circle of the whole circle equals:

$$\delta_\mathrm{E} = \frac{N_\mathrm{g}(d)}{N_\mathrm{g}} = \left(\frac{d}{r_\mathrm{g}}\right)^2 \propto d^2 \tag{4.9}$$

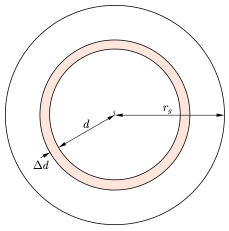


Fig.7 An example of observing from the ground

Now we need to calculate the amount of entities within a Δ_d ring, whose proportion can be calculated using the formulae below

$$\Delta_d = N_g(d+\Delta_d) - N_g(d) = \frac{1}{{r_g}^2}\left[(d+\Delta_d)^2 - d^2\right] \xrightarrow{\Delta_d \to 0} \frac{2}{r_g^2}\Delta_d \tag{4.10}$$

So the amount of pixels to the viewer's eye (according to section 4.5) equals:

$$\frac{2\Delta_d}{r_g} N_g \times P_{\text{Target}} = 2\Delta_{d_E} L_E C \left(\frac{r-d}{r-1}\right)^2 \tag{4.11}$$

Now we need to calculate the amount of all pixels in the circle to the viewer's eye. In order to achieve this, we sum all the pixels required together and get the following definition formula and the results are shown int he diagram below.

$$P_{\text{reload}} = \lim_{\Delta d \to 0} \sum_{i=1}^{\left\lfloor \frac{r_g}{\Delta d} \right\rfloor} \left[\frac{2\Delta d}{r_g} \cdot N_g \cdot 2\Delta d_E L_E C \cdot \left(\frac{r_g - i\Delta d}{r_g - 1}\right)^2 \right]$$

$$\xrightarrow{\text{let } A \text{ equal } \frac{4N_g \Delta d_E L_E C}{r_g} = 4\pi r_g \rho_E \Delta d_E L_E C} A \cdot \lim_{\Delta d \to 0} \sum_{i=1}^{\left\lfloor \frac{r_g}{\Delta d} \right\rfloor} \Delta d \cdot \left(\frac{r_g - i\Delta d}{r_g - 1}\right)^2$$

$$= A \cdot \int_0^{r_g} \left(\frac{r_g - x}{r_g - 1}\right)^2 \mathrm{d}x$$

$$= \frac{4\pi r_g^4 \rho_E \Delta d_E L_E C}{3(r_g - 1)^2}$$

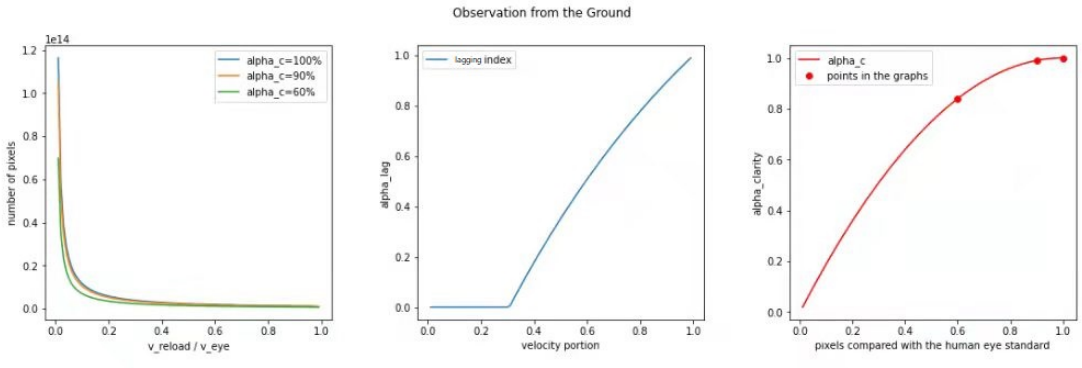


Fig.n Patterns of x, α_{lagging}, α_c from the ground

4.5.2 Observing From the Air

Another way, which is commonly seen in various type of games, is having a bird's eye view of the ground. Since there are no obstacles from the air, the observing range is likely to be significantly higher. According to official data, when man observe from the sky, the average observing range r_a is about 13 kilometers.

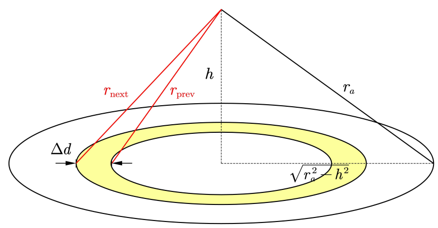


Fig.10 An example of observing from the air

For entities in the marked part, for $\Delta_d \to 0$ ($\Delta_d << h$), we take r_{next} as the distance from the observer to all the entities in the ring. So we can get the formula:

$$\text{Number of pixels within a } \Delta d \text{ range} = \frac{2\Delta d}{\sqrt{r_a^2 - h^2}} \cdot N_g\left(\sqrt{r_a^2 - h^2}\right) \cdot 2\Delta d_E L_E C \cdot \left(\frac{r_a - r_{next}}{r_a - 1}\right)^2$$

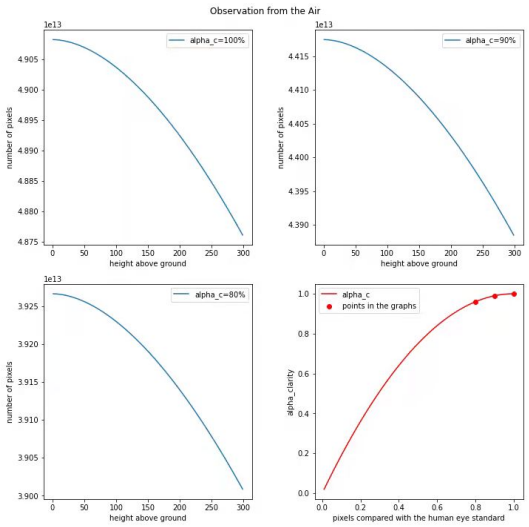
$$= \frac{4\pi\sqrt{r_a^2 - h^2}\,\rho_E \Delta d_E L_E C}{(r_a - 1)^2} \cdot \Delta d\,(r_a - r_{next})^2, \text{ where } r_{next} = \sqrt{h^2 + (i\,\Delta d)^2}$$

$$\xlongequal[\quad]{B = \frac{4\pi\sqrt{r_a^2 - h^2}\,\rho_E \Delta d_E L_E C}{(r_a - 1)^2} \text{ for short}} B \cdot \Delta d\,(r_a - r_{next})^2$$

$$r_{next} = \sqrt{h^2 + (i\Delta_d)} \tag{4.12}$$

Where i is the number of layer of the ring from the inside out, we can get a summarized formula. The result is shown in the following diagram.

$$
\begin{aligned}
P_{\text{reload}} &= \lim_{\Delta d \to 0} \sum_{i=1}^{\left\lfloor \frac{\sqrt{r_a^2 - h^2}}{\Delta d} \right\rfloor} B\,\Delta d \cdot \left(r_a - \sqrt{h^2 + (i\,\Delta d)^2}\right) \\
&= \int_0^{\sqrt{r_a^2 - h^2}} B \cdot \left(r_a - \sqrt{h^2 + x^2}\right) \mathrm{d}x \\
&= B \left[x(h^2 + r_a^2) + r_a \sqrt{h^2 + x^2} \cdot \left(- \frac{h \cdot \sinh^{-1}\left(\frac{x}{h}\right)}{\sqrt{\left(\frac{x}{h}\right)^2 + 1}} - x \right) + \frac{x^3}{3} \right] \Bigg|_0^{\sqrt{r_a^2 - h^2}} \\
&= B \left[\sqrt{r_a^2 - h^2}\,(h^2 + r_a^2) - r_a^2 \cdot \left(\frac{h \cdot \sinh^{-1}\left(\frac{\sqrt{r_a^2 - h^2}}{h}\right)}{\frac{r_a}{h}} + \sqrt{r_a^2 - h^2} \right) + \frac{(r_a^2 - h^2)^{\frac{3}{2}}}{3} + r_a h^2 \sinh^{-1}(0) \right] \\
&= \frac{4\pi \sqrt{r_a^2 - h^2}\,\rho_E \Delta d_E L_E C}{(r_a - 1)^2} \cdot \left(\sqrt{r_a^2 - h^2} \cdot h^2 - h^2 \cdot r_a \sinh^{-1}\left(\frac{\sqrt{r_a^2 - h^2}}{h}\right) + \frac{(r_a^2 - h^2)^{\frac{3}{2}}}{3} \right)
\end{aligned}
$$



Patterns of α_c when observing from the air

4.5.3 Observing From Planes

The third condition is observing from planes, we abstract it as a regional calculation on the ground. The average observation range of human height is far less than r_a under this circumstance. We find out the following formula:

$$
P_{\text{Reload}} = \frac{4\pi L_{\text{airport}}^4 \Delta_{d_E} L_E C}{3\left(L_{\text{airport}} - 1\right)^2} \tag{4.13}
$$

According to the data online, we find that L_{airport} is about 3452.5979 meter. So we find out that :

$$V_{\text{plane}} \approx 900\text{m/s} \tag{4.14}$$

$$L_{\text{plane}} \sqrt[3]{58.82 \times 12.61 \times 17.39} \approx 23.452\text{m} \tag{4.15}$$

$$\rho_{\text{plane}} = 0.163 \times 10^{-}6\text{m}^{-}2 \tag{4.16}$$

$$(r_{\text{g}} \approx r_{\text{a}} >> L_{\text{airport}}) \tag{4.17}$$

We can similarly get the data and do the same calculations on the underground. Here are the results:

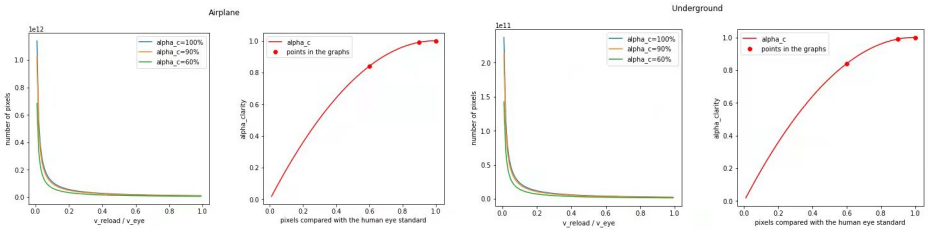


Fig.12&13 Patterns of x, α_{c} of observing planes and underground

4.5.4 Observing From Roads

Finally, we will consider the condition where observers walk on streets. To facilitate our calculation, we see the streets as straight lines which have width shown in the figure below.

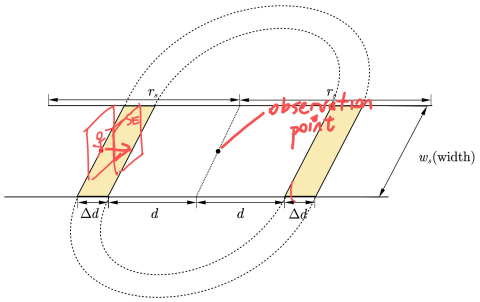


Fig.11 An example of observing from roads

From this picture, we can figure out the relationship of P_{reload} with other determining factors in the following formula:

$$P_{\text{reload}} = \lim_{\Delta d \to 0} \sum_{i=1}^{\lfloor \frac{r_s}{\Delta d} \rfloor} \rho_E \Delta d \cdot W_s \cdot C \cdot \left(\frac{r_s - i\Delta d}{r_s - 1} \right)^2 \cdot \frac{S_E}{L_E} \cdot 2\Delta d_E$$

$$= \frac{2\rho_E W_s C S_E \Delta d_E}{(r_s - 1)^2 L_E} \int_0^{r_s} x\,(r_s - x)^2 \, \mathrm{d}x$$

$$= \frac{r_s^4 \rho_E W_s C S_E \Delta d_E}{6(r_s-1)^2 L_E}$$

4.6 Calculating Storage Require

When we calculate the byte required to store this virtual city, we also need to find out how many times it run in one second.

$$P_{\text{calc\_cloud}} = V_{\text{reload}} \cdot u_{\text{E}} \cdot \left(2L_{\text{E}}^2 \cdot \Delta_{d_{\text{E}}}\right) \tag{4.18}$$

$$P_{\text{calc\_local}} = V_{\text{reload}} \cdot \Sigma_{E \in entity} \left(\frac{r-d}{r-1}\right)^2 L_{\text{E}} C \tag{4.19}$$

Then we need to calculate how much storage space it requires to maintain this virtual city (S(storage)), it is divided into S_{cloud} and S_{local}. We can find the following formula:

$$S_{\text{cloud}} = \Sigma_{\text{number of entity types}} \text{number of parameters} \cdot \text{number ofentities} \tag{4.20}$$

$$S_{\text{local}} = \Sigma_{E \in entity} N_{\text{E}} = k\Sigma_{E \in entity} \left(\frac{r-d}{r-1}\right)^2 L_{\text{E}} C \tag{4.21}$$

To further evaluate the storage level of virtual city, we introduced a comprehensive index to describe it. It's related to α_{c}, α_{lagging}, α_{fit}, α_{reload}. Their weights are shown below.

$$K \overset{def}{=\!=} \omega_1 \alpha_{\text{c}} + \omega_2 \alpha_{\text{lagging}} + \omega_3 \alpha_{\text{fit}} + \omega_4 \alpha_{\text{fitting}} \tag{4.22}$$

We have to note that $0 < \omega_1,\ \omega_2,\ \omega_3,\ \omega_4 < 1, \sum \omega_1 = 1$.

Though we have constructed a virtual Hong Kong in *metaverse*, there are some problems that we have not solved.As we know, the entities we see in real world is not continuous (for example, it' s impossible for you to see exactly 0.1 car every second). Because of this, we can calculate the proportion of time when we can see a kind of entity in a day to optimize the division of storage.

For ferries, the time for each ferry to be seen is equal to the visibility divided by the speed (100s) while the average time interval of department is 175.15s. So the visible proportion of ferries is 57.10%. With the same method, we can calculate that the proportion of planes is 8.10% and the proportion of subway is 13.33%.

type	ferry	airplane	underground	vehicles	people
number of pixels	2.06E+12	9.2E+10	2.8E+10	3.7E+9	3.6E+9

Sum these numbers up and we get S_{local} = 255GB. In addition, S_{cloud} = 31.9TB. By the way, the result before the optimization is 14.2 TB, the optimizing process works well. Also, by the previous formulae we can calculate that $P_{\text{calc\_local}}$ = 1.75E + 13 and $P_{\text{calc\_cloud}}$ = 1.58E + 14 per second.

5 Strengths and Weaknesses

Strengths

- **Authenticity**

  All the data mentioned in our model is provided by authority in Hong Kong, so it can simulate the real Hong Kong well.

- **Simplicity**

  All the variables in our model are simple to understand, so there isn't much calculation needed and is easy to handle.

Weaknesses

- **Inaccuracy**

  Due to our assumptions, we can not describe the shape of blocks accurately. This may lead to inaccuracy of the whole virtual city.

- **Non-real Time**

  There are many kinds of observation angles in our model, so we can not realize the real time upload in our model.

References

[1] EPSG:2326 Hong Kong 1980 Grid System, MapTiler Team
https://epsg.io/2326

[2] https://www.map.gov.hk/gm/map，GeoInfo Map，

[3] https://www.flightradar24.com/airport/hkg，Hong Kong International Airport，FlightRadar24，

[4] https://en.wikipedia.org/wiki/MTR，Hong Kong MTR，Wikipedia，在火车数的地方引用