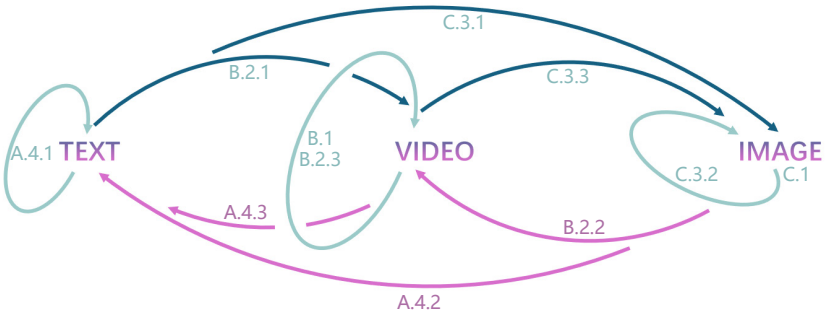
SKILLS CLASS RC11 23/24
STUDENT NUMBER: 23121108

VECTORIAL ENCODINGS OF QUALITATIVE DOMAINS

FINAL ASSIGNMENT

Onedrive_Link: https://liveuclac-my.sharepoint.com/:u:/g/personal/ucbvtw1_ucl_ac_uk/EYRxSqJ3K1RFvyIH2RLcyJABK472IJCJ11gzv9
Github_Link: https://github.com/UD-Skills-2023-24/23121108/blob/main/Julian/RC11_23121108_FINALASSIGNMENT.ipynb
Colab_Link: https://colab.research.google.com/drive/1lf-UzfYToFc9hzCxDpT7UvM0CVH459YM?usp=sharing



Intro

In this assignment, I mainly collected three datasets: Spanish novels, old photos of Madrid and Almodovar films. I vectorized them in different ways, so that every two data sets could be retrieved from each other.

I try to vectorize text using three vectorized methods. Then I picked one of them to train SOM. For the picture database, on the one hand, I directly vectorize the picture, and each picture can be given a 1024 dimension feature. On the other hand, I also extracted the description of the picture through the fuyu model. For the video database I mainly vectorized the captions. Then use subtitles to position the timeline to get a slice of the video.

1 Text SOM Training and Search

1.1 LSA VS TF-IDF

It can be seen from the output result that LSA can capture the semantic correlation between words, even if these words do not directly co-occur in the corpus. TF-IDF is mainly based on word frequency and inverse document frequency, and ignores the semantic correlation between words, which may cause some words with weak correlation but similar semantics to be ignored. In addition, I found that neither LSA nor TF-IDF could handle very sparse matrices. But Lsas seem to handle better. Perhaps Lsas can extract commonalities between documents through singular value decomposition, thereby reducing differences between vector representations.

```
[{'paragraph': ' When Mr Colpits' s face showed that he was going to make a remark about that, Matthew was before
him: 'We' re researchers. We' re getting a pass from the French government.' The first annoyance. For a time no
one could talk to Matthew, and everyone stayed out of his way: scientists, midshipmen and cats, even the cook.',
  'nr': 467,
  'ID': 'The Discovery Of Slowness (Nadolny Sten) (Z-Library)',
  'type': 'epub'},
 {'paragraph': ' After three days their faces became even longer. 'Now she takes in four inches per hour,' the
First Lieutenant said. 'Soon we' ll need no cats. The rats will drown on their own.' * * * Madeira! John was on
land again.',
  'nr': 531,
  'ID': 'The Discovery Of Slowness (Nadolny Sten) (Z-Library)',
  'type': 'epub'},
```

ouput of LSA

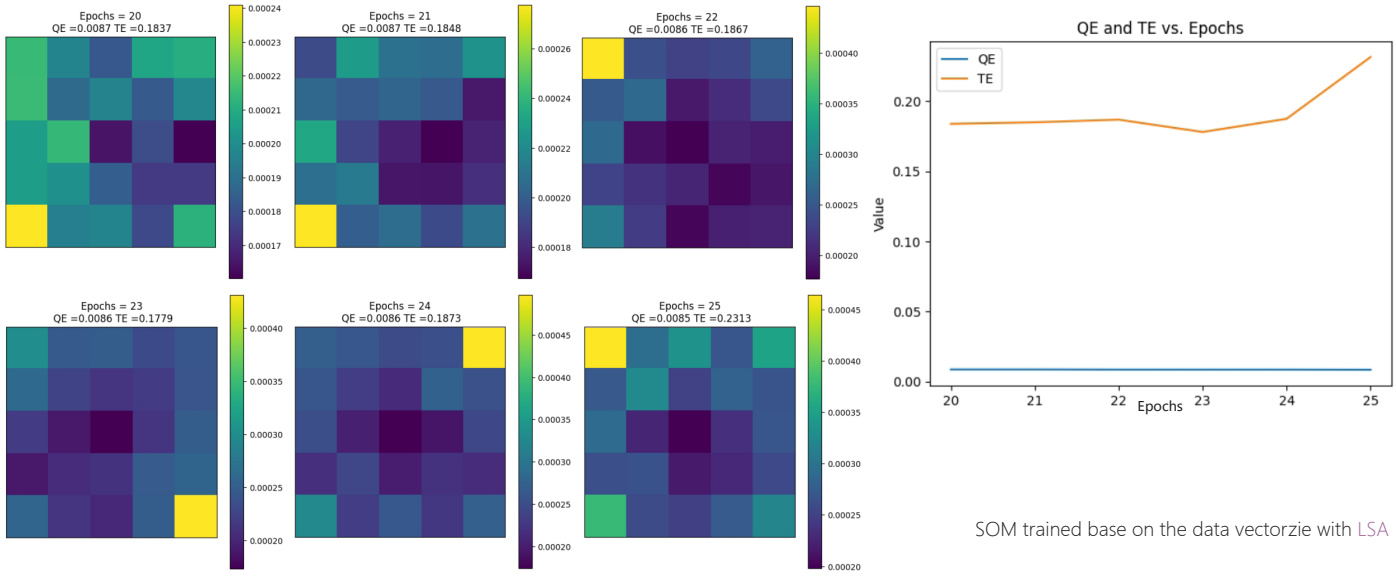
```
[{'paragraph': " Once I looked around mid-lesson and could not find Alberto at all. I remembered that he had been
waving at me, calling out to me about dynamics, but now I could not find him. The balcony was empty. I hadn't
heard the front door close. Yet it had—with a slam, he told me later. In a pique, he had left for the café.",
  'nr': 344,
  'ID': 'The Spanish Bow (Romano-Lax, Andromeda) (Z-Library)',
  'type': 'epub'},
 {'paragraph': ' "Besides," I said, searching the rumpled bedclothes for my trousers, "she\'s not likely to find a
husband. There are complicating factors." "Previously married?" "No." "Ruined reputation? Non-Catholic?"',
  'nr': 1673,
  'ID': 'The Spanish Bow (Romano-Lax, Andromeda) (Z-Library)',
  'type': 'epub'},
 {'paragraph': ' 'It' 11 do them harm in any event,' Back suggested casually. 'Let' s just hope they don' t find
out while we depend on them.' The briefest sentence of all was spoken by Hepburn, a Scotsman from near
Edinburgh.',
```

ouput of TF-IDF

Code directory

1.2 Train SOM

I write a function that recorded the QE and TE of each epochs and plotted it as a line graph. TE mainly affects the topological structure and spatial distribution of data. QE reßects the ability of SOM model to represent input data. Reducing QE helps SOM model to represent input data more accurately.Both QE and TE are about as small as possible. QE has changed relatively little here. We chose SOM with the lowest TE. DeÞne SOM = soms[2] for subsequent processing.

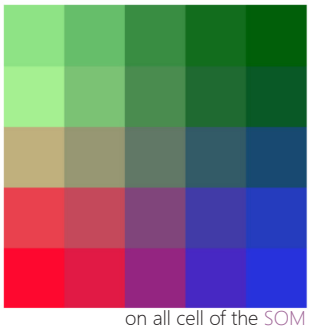


SOM trained base on the data vectorzie with LSA

1.3.1 Fit PCA on all cells of the SOM

Principal Component Analysis (PCA) is a commonly used dimensionality reduction technique for converting a high-dimensional data set into a low-dimensional subspace while retaining most of the information in the data set.

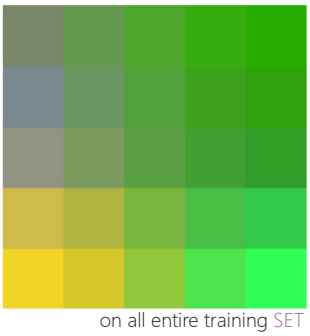
The vector of each cell in LSA som is downscaled to 3 dimensions using PCA and the three dimensions are assigned to that individual unit as RGB corresponding to the colour. It finall showed the som as this.



on all cell of the SOM

1.3.2 Fit PCA on the entire training set

We can clearly see that the color difference in the above picture is quite large. Because when Fit PCA on all cells of the SOM. The spatial structure between units and the vector features behind them are not reduced until the organization is complete. In the Þgure below, all of the cells are neighboring colors. After reducing the dimensions of all the data, the characteristics of the data will become more consistent. So the characteristic structure expressed by different neurons in all the obtained SOM images is more consistent, and the whole is more unif ied.



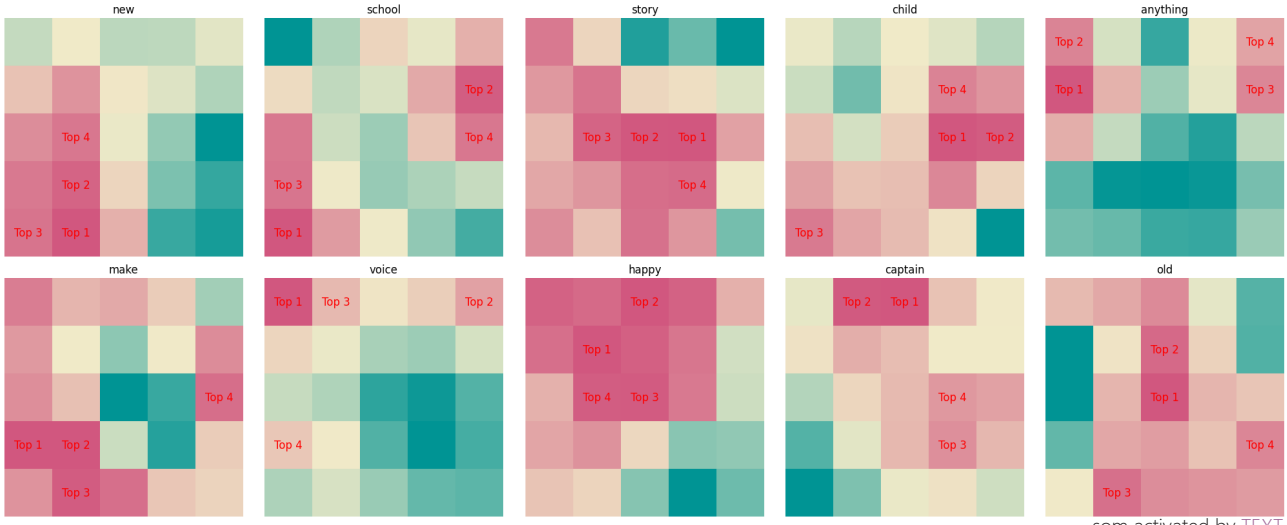
on all entire training SET

1.4 Ativate SOM

For each SOM, I set up different activation methods for different types of input data.

1.4.1 Input text

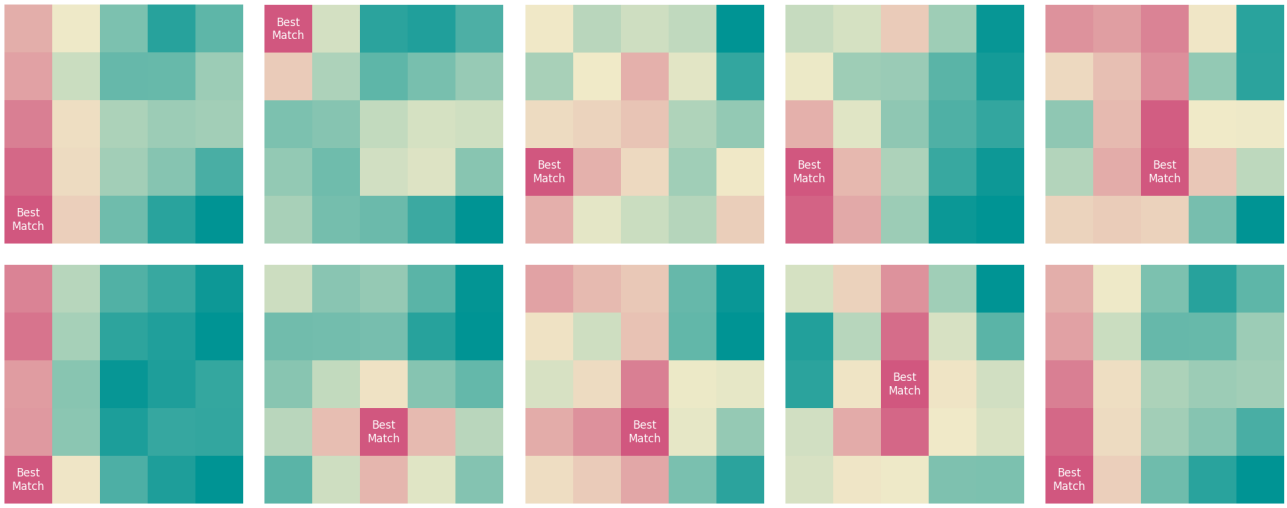
For input located in text, we first vectorize it in the same way as we do with training data. And then calculate the cosine distance from each neuron. In this way, the activated neurons and their corresponding data can be obtained.



som activated by TEXT

1.4.2 Input image

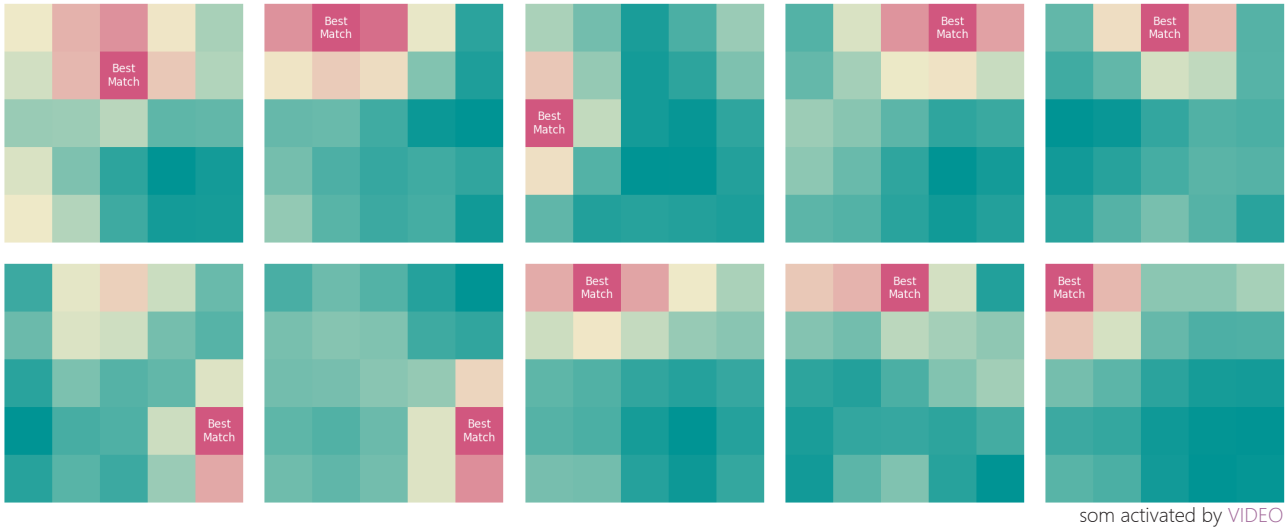
It is difÞcult to directly compare vectorized features of images and text. So I preprocessed all the images. Store the id of the image and description of the contents in a csv Þle. Here I call ten descriptions randomly to activate text SOM.



som activated by IMAGE

1.4.3 Input video

We can use the fuyu model to get a description of all the frames of the movie or use subtitles to video activate the text SOM. Here I chose the second option.



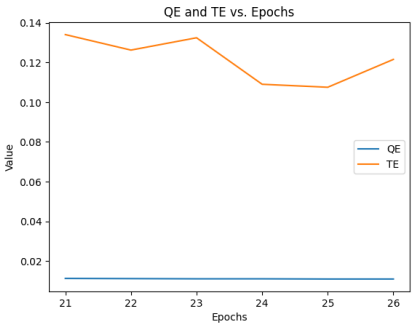
som activated by VIDEO

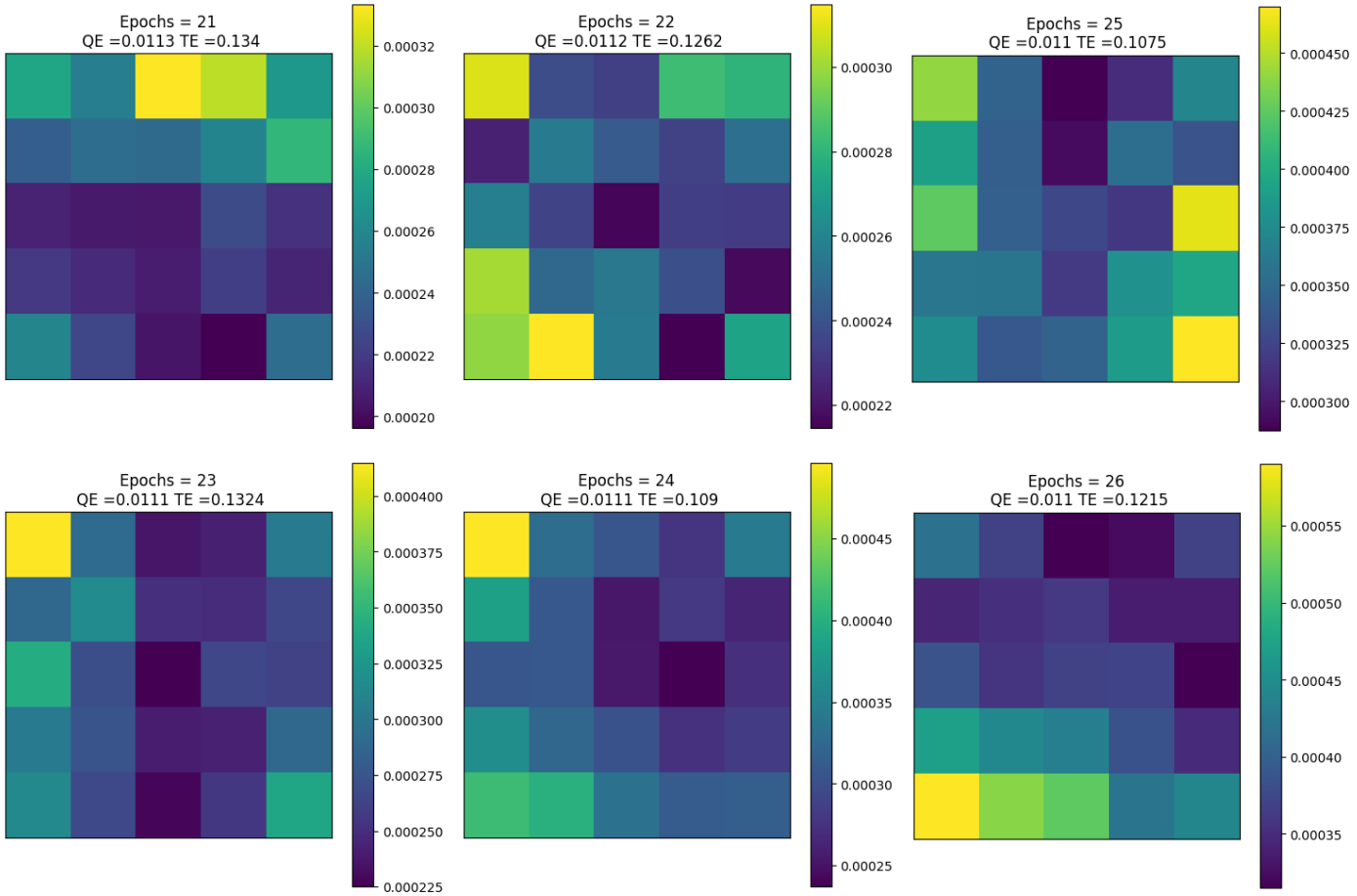
2 Video SOM Training and Search

2.1 SOM training

Video SOM's training was based on my collection of Almodovar's films. All subtitle files are preprocessed and their timestamps, frame sequences, and corresponding subtitle content are stored in a dictionary. It is convenient to call at any time and locate the timeline in the later period.

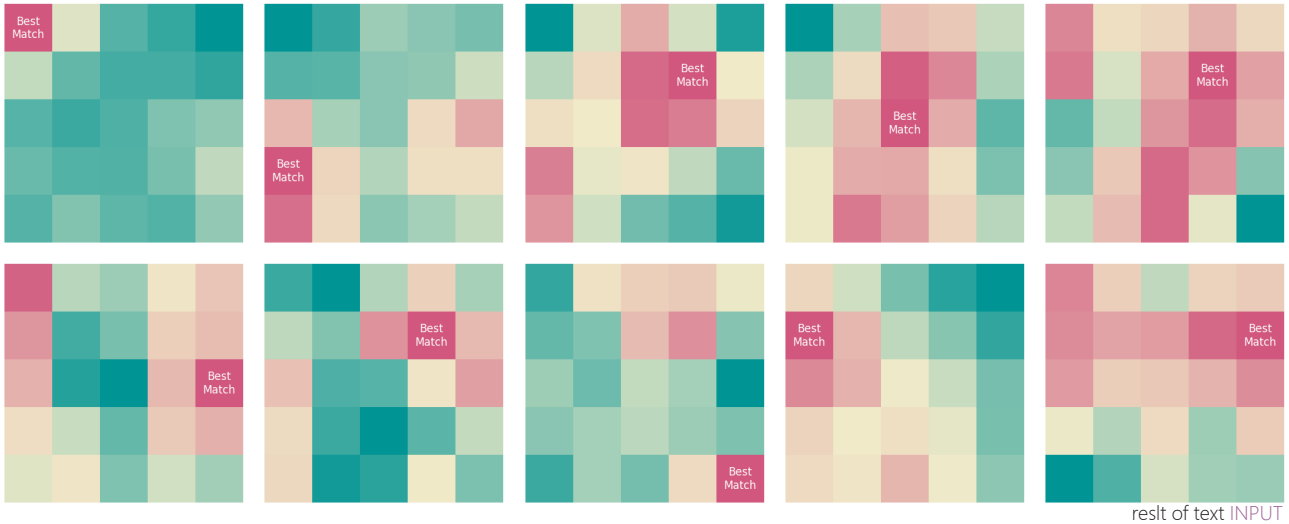
All of my trained SOMs are stored in dictionaries and saved as pickle Þles. It can be invoked at any time during later activation





2.2 SOM activate

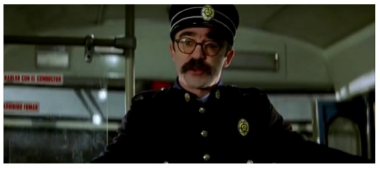
Since my training set is a subtitle Þle. All three different inputs are essentially similar to the activation of the text SOM just mentioned. So here I mainly show the output results. When we activate SOM, we get the most similar subtitle snippets. Through the content of the subtitle clip we can re-index the corresponding time stamp and extract the movie clip from it.



reslt of text INPUT

Best Match - Index: (0, 0), Content: Who have we got here? Will you give Grandpa a kiss? - How was the trip? - Really long. We're worn out. Let me introduce you. This is Sanáa. Julieta. And this wonder is Antía. Have you got a case? Yes, a brown one, it's got my name on it. Can you get it? And a stroller! How's Mom? Fine, we're getting by. You'll see her now. And that girl? It's Sanáa, the girl I told you about. She's the woman who helps you? She looks after your mom and helps me on the land. We were lucky to Þnd her. She's changed our lives. Let's get in the car. - Don't you miss the school? - I don't have time. Between your mother and the land,

Best Match - Index: (3, 0), Content: That's no good, because I'm the teacher and teachers aren't supposed to have sex with their pupils. - Tell me someone else. - Kim Basinger. Really lovely. Much prettier than me, toady! Ã□ngela Molina. She's beautiful, too. Well, Calypso was as beautiful as Kim Basinger and Ã□ngela Molina together. And she offered Ulysses everything you could imagine. Diego, what could she have offered him? Her body. Of course. And what else? Something we've all dreamed of... - Eternal youth. - Exactly. And immortality. Yet Ulysses rejected it and he set sail to face countless dangers. Which of the three meanings would you choose



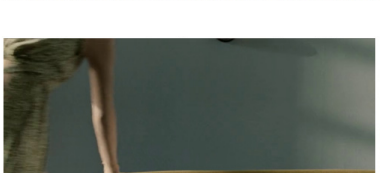
Best Match - Index: (3, 0), Content: That's no good, because I'm the teacher and teachers aren't supposed to have sex with their pupils. - Tell me someone else. - Kim Basinger. Really lovely. Much prettier than me, toady! Ã□ngela Molina. She's beautiful, too. Well, Calypso was as beautiful as Kim Basinger and Ã□ngela Molina together. And she offered Ulysses everything you could imagine. Diego, what could she have offered him? Her body. Of course. And what else? Something we've all dreamed of... - Eternal youth. - Exactly. And immortality. Yet Ulysses rejected it and he set sail to face countless dangers. Which of the three meanings would you choose

Best Match - Index: (2, 2), Content: I noticed him because he didn't have a coat. I think he was drunk. He's probably still wandering around. That's why we've stopped. Take this. Get back inside! You'll catch pneumonia. What's on the stretcher? What? Is it the stag? No, why do you say that? Is it a man? Go inside. Is it him? Is that why we stopped? A man in a black sweater. He had glasses. Please, get on the train. He was sitting where you are now. He wanted to talk, but I didn't like how he was looking at me and I ran out. How could I have known he felt so bad? Any girl would have done the same. I should have realized. Don't torture yourself.

Best Match - Index: (1, 3), Content: I got your letter and I came. I never imagined that your wife... Don't think about that now. Good morning. - These cookies are so good! - They're typical of here. I was Þxing breakfast. I was really hungry. Anything for me? We'll Þnd something... I took your letter as an invitation to come and see you. That was my intention. I never thought that your wife had died. I'm sorry. It was for the best. She had no life. Marian told me you were here, waiting for the bus, but that she wasn't certain you'd leave. She told me you were with a woman. Ava, I think. She's an old friend, you'll meet her. She's an artist like you. I'm not an artist. I teach Classical Literature. And now, not even that.

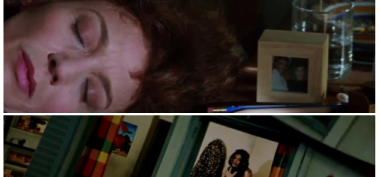
Best Match - Index: (2, 4), Content: We'll get there early. You don't have to, Mom. I'm with my friend Bea. I met her here and now we're best friends. Can I go and spend a week with them in Madrid? You don't mind, do you? But we don't know them. But she's my friend... I'll put her mother on, she can tell you. Hello, how are you? I'm Claudia, Beatriz's mother. How's my daughter? Very well, great. I didn't know her before, but camp has done wonders for her. I wanted to ask your permission to take her to Madrid with us. They've grown really close and are making such a fuss about it... Doesn't the camp end tomorrow? Yes, but I came a day early. I was hoping to take them this afternoon. What do you think? Well, if she'd like that...

Best Match - Index: (1, 3), Content: I got your letter and I came. I never imagined that your wife... Don't think about that now. Good morning. - These cookies are so good! - They're typical of here. I was Þxing breakfast. I was really hungry. Anything for me? We'll Þnd something... I took your letter as an invitation to come and see you. That was my intention. I never thought that your wife had died. I'm sorry. It was for the best. She had no life. Marian told me you were here, waiting for the bus, but that she wasn't certain you'd leave. She told me you were with a woman. Ava, I think. She's an old friend, you'll meet her. She's an artist like you. I'm not an artist. I teach Classical Literature. And now, not even that.

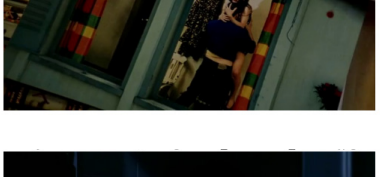


Best Match - Index: (4, 4), Content: because you were a child, because it was too painful for me, or simply out of shame. But you're not a child anymore. Beatriz told me that you have children of your own. Three, no less. You're a grown woman, and a mother. Where do I begin? I'll tell you about your father. When you asked me how I met him, I told you it was on a train. But I didn't tell you everything. I was 25. It was an unpleasant night and very windy. Is this seat free? Yes. That branch was really something. Did it scare you? Yes. Are you traveling alone? Yes. When I saw you alone I thought, "We'll keep each other company. "

Best Match - Index: (1, 0), Content: Come on, Canelo! Leave it there. We'll go into the kitchen. When did you meet Ana? Ana? I didn't know her. I supposed so. She was in bed the last six years. The sea! It's really striking the Þrst time. Sit down. So you've come to see Xoan. Do you want a cookie? No. I don't think he'll be back tonight. He's with Ava. Do you know Ava? She does ceramics and sculptures. Ana and Ava were good friends, when Ana could still talk or listen. The poor thing was a vegetable, and a man needs a woman. But it's not for me to criticize Xoan. More coffee? - No. - Then give me the cup. So what are you going to do?



Best Match - Index: (1, 4), Content: for the sea that Ulysses yearned for? Thalassa? No! - Pontos. - That's it! Pontos, the sea, the high sea, the road to adventure and the unknown. Come in. Hello. Julieta, have a seat. Thank you. Mrs. Martínez is coming back next week so your substitution time is over. We are very happy with your work. Very happy indeed. There's never been such attendance in the Classical Literature classes. Thank you. I've really enjoyed these six months. I'm delighted. Well... Wait. This letter came, it's for you. Is it for you? Yes, it is. Dear Julieta,



subtitle of the corresponding video CLIP

screenshot of the corresponding video CLIP

3 Image SOM Training and Search

3.1 SOM training

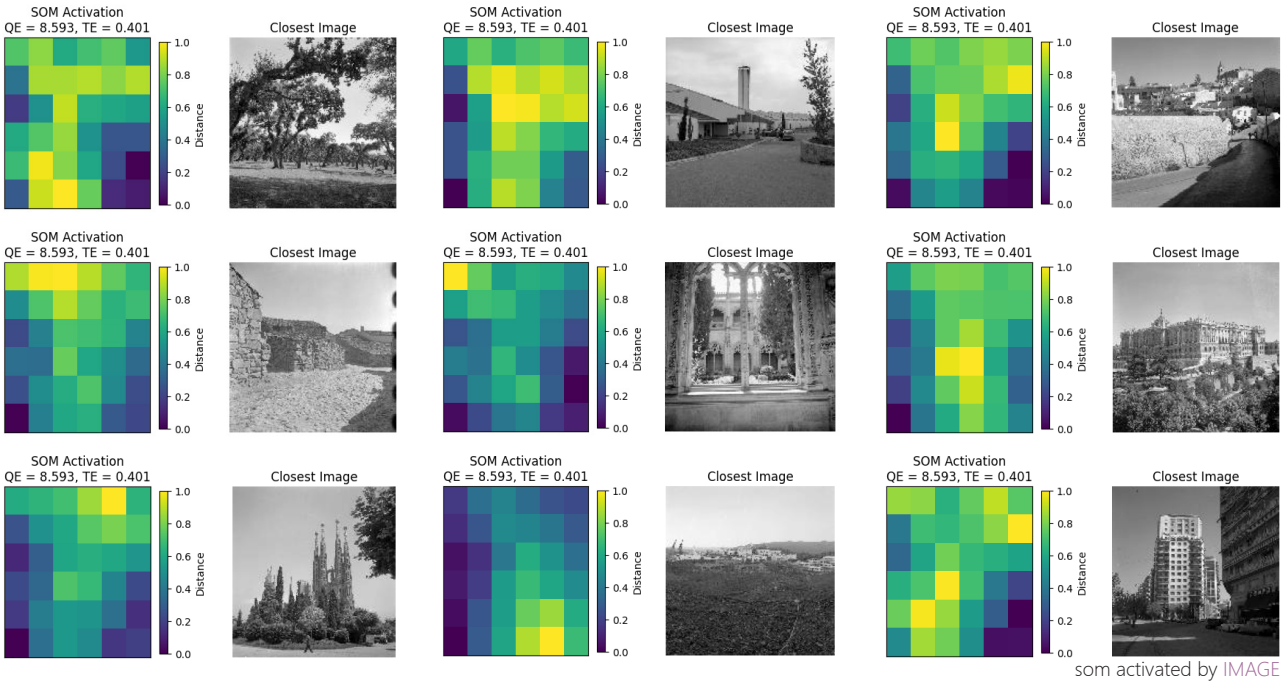
I prepared two types of SOM for the image dataset. I use the image vector to 1024 dimensions of data. And the picture with the highest score of each unit is projected on the corresponding unit. This SOM is primarily used for image input activation. I think it's intuitive. For text and video input. I used the picture descriptions that fuyu got as training data.

map the SOM

3.2 SOM activate

3.2.1 Input image

Images can be entered directly and the most matching images can be obtained.

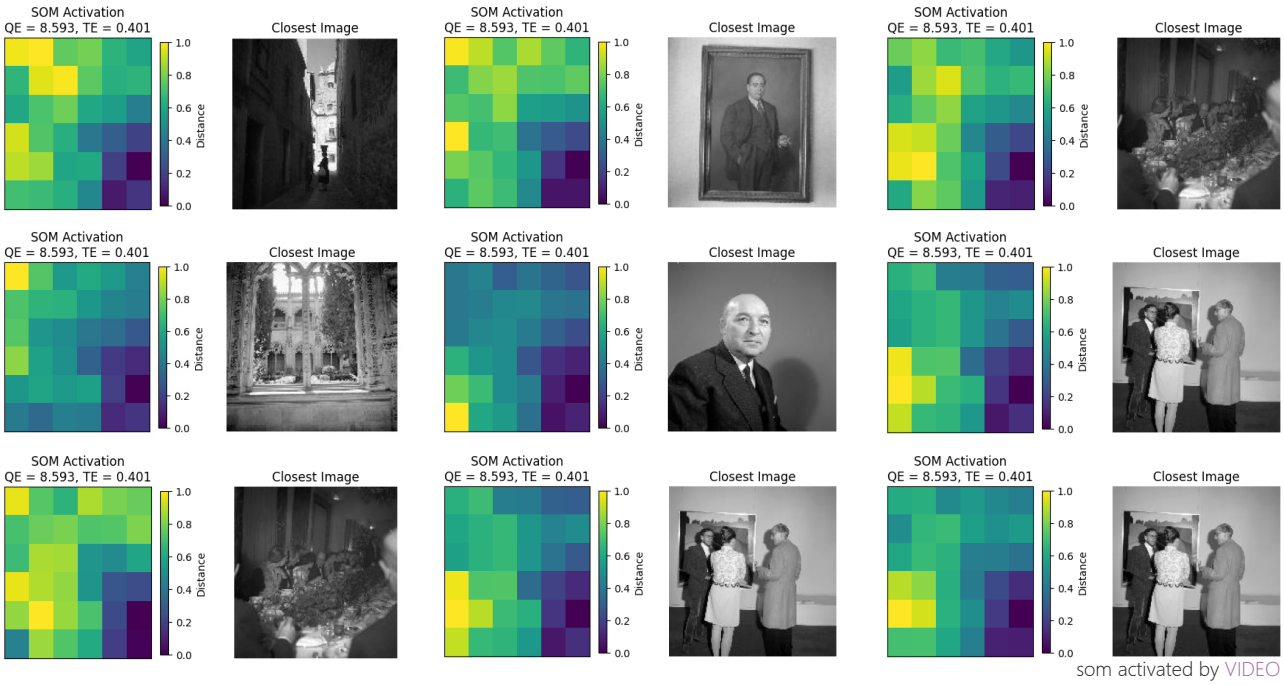


som activated by IMAGE

3.2.2 Input text and video

For text and video (subtitle) input, I redeÞned a class TextModel2 that reads the description of the image in the csv and vectorizes it. Using this data, another SOM was trained to interpret the video and text input.

```
class TextModel2:
    def __init__(self, files, vectorization='lsa', dimension=200, min_df=2):
        self.vectorization = vectorization
        self.paragraphs = []

        for f in files:
            ID = os.path.splitext(os.path.basename(f))[0]
            filetype = f.split('.')[-1]
            if filetype == 'csv':
                paragraph = read_csv_paragraphs(f, ID, 'csv')

            self.paragraphs.extend(paragraph)

        self.preprocessed_paragraphs = preprocess(p['paragraph'] for p in self.paragraphs)

        if self.vectorization == 'tfidf':
            self.tfidf_vectorizer = TfidfVectorizer(min_df=min_df)
            self.vector_matrix = self.tfidf_vectorizer.fit_transform(self.preprocessed_paragraphs)
        elif self.vectorization == 'lsa':
            self.tfidf_vectorizer = TfidfVectorizer(min_df=min_df)
            self.tfidf_matrix = self.tfidf_vectorizer.fit_transform(self.preprocessed_paragraphs)
            self.svd = TruncatedSVD(n_components=dimension, algorithm='randomized')
            self.vector_matrix = self.svd.fit_transform(self.tfidf_matrix)
        self.nnModel = NearestNeighbors(n_neighbors=10,
                                        metric='cosine',
                                        algorithm='brute',
                                        n_jobs=-1)
        self.nnModel.fit(self.vector_matrix)

    def vectorize(self, query):
        if self.vectorization == 'lsa':
            processedQuery = preprocess([query])[0]
            tfidf_query = self.tfidf_vectorizer.transform([processedQuery])
            query_vector = self.svd.transform(tfidf_query)
            return query_vector
        elif self.vectorization == 'tfidf':
            processedQuery = preprocess([query])[0]
            query_vector = self.tfidf_vectorizer.transform([processedQuery])
            return query_vector

    def search(self, query, n=3, distance=False):
        qv = self.vectorize(query)
        neighbours = self.nnModel.kneighbors(qv, n, return_distance=distance)[0]
        paragraphs = [tuple(p.items()) for p in (self.paragraphs[i] for i in neighbours)]
        unique_paragraphs = list(set(paragraphs))
        return [dict(p) for p in unique_paragraphs]

    def get_key_words(self, v, n=10):
        if self.vectorization == 'lsa':
            v = self.svd.inverse_transform(v)[0]
            top_indices = np.argpartition(v, -n)[-n:]
            words = self.tfidf_vectorizer.get_feature_names_out()
            return [words[i] for i in top_indices]
        elif self.vectorization == 'tfidf':
            top_indices = np.argpartition(v, -n)[-n:]
            words = self.tfidf_vectorizer.get_feature_names_out()
            return [words[i] for i in top_indices]
```



som activated by VIDEO

4 Example of the Work Flow

Here I will show the workßow I implemented using this search engine in my project. I start with the text and use the get_keyword function to get 10 keywords. Activate text SOM as input to get 10 relevant paragraphs.

```
[92] # Get initial input
paragraph_values = novel_model.get_key_words(novel_model.vectorize('Chivalry never dies, even though the armor will rust one day, the sword will always be sharp'), n=10)
print(paragraph_values)
```

['changed', 'tom', 'closer', 'one', 'never', 'kept', 'even', 'always', 'though', 'day']

```
[93] # This list will as input of next step
paragraph_values = []

for paragraph in paragraphs_list:
        results = novel_model.search(paragraph, n=1)
        for result in results:
                paragraph_value = result['paragraph']
                paragraph_values.append(paragraph_value)

print("Paragraph Values:", paragraph_values)
```

Paragraph Values: [' Mockridge thought of other things. 'Portsmouth,' he mused. 'I know a lot of girls there.' The eye geared to distances had already focused Þrmly on girls. Stanley Kirkeby agreed and declared that this was just what the doctor ordered.', ' 'How was it?' John reßected for a moment, then made a decision and stuck to it. 'I'm in love,' he said. 'Only I was a bit discouraged at Þrst because of the buttons.' That was no lie, really. For a long time the pleasant scent of her skin lingered in his mind.', ' Noticing my raised eyebrows, he added, "City of Bombs—you know that\'s what they call Barcelona? At least once a year the Ramblas is Þlled with smoke. Between that and church burnings, the city is reborn. It isn\'t a bad thing, necessarily. There are some nice public plazas where cathedrals once stood."', ' "But I\'m wasting your talent," she said. "You shouldn\'t be giving anyone Spanish lessons. You should be giving music lessons." "Do you play?" I said, too eagerly. "I don\'t." She read my disappointment. "I used to. But I wasn\'t very good. And where I grew up, silence was the rule, especially on rainy days, when we all crowded indoors. \'Children should be seen, not heard\'—perhaps you have heard that English expression."', ' Do you know the painter Westall?'Babbage passed over the question. 'For a sailor you think pretty fast,' he said in a subdued voice.'No, I think with an effort,' Franklin replied, 'but I never stop. You know too few sailors.', ' One of Akaitcho's warriors stayed behind, too – for her. Everyone except Hood knew this. Even John.Hood talked of returning at the end of the journey to live with Green Stockings in Fort Providence, or wherever.', 'Eleanor died a few days after John Franklin and his companions had boarded the ship in Liverpool. He heard the news months later in Canada, after he had written her several comforting and cheering letters.', " Now there were Nationalists both north and south, planning to meet in the middle. Franco made an appointment with a foreign reporter to meet him at a certain table in a certain Madrid café; he was that sure he'd be taking the capital according to his own timetable.", " About Wentzel, Back said, 'Now that's a real German. Everywhere in the world you see them standing around brooding about why they can't get a move on like anyone else. And most of the time they try to prove that the reason is their intelligence, and then they start to teach the rest of mankind.', ' "Forgive me," I said. "I can\'t say I understand this." How could I explain that everyone knew about the King\'s inÞdelities? I couldn\'t see how one more tryst would tip the moral scales. She glared at me irritably. "The Church is refusing the new tax. It\'s no small sum. The extreme clericals are stronger than ever, and the liberals are losing patience. The King can\'t please them both. I\'m not sure he can please anyone." After a deep breath, she continued. "The duchess and her husband are vocal opponents of the new tax. If the King indulges her, people will make assumptions—perhaps correct ones. If he chooses that path, well ... I hear Gibraltar has become crowded since the Portuguese Republicans took Lisbon."]

Then input these paragraphs into picture SOM to get similar description and corresponding pictures with these ten description. I noticed that many duplicate units were activated. This may be because the initial input was 10 words with similar features.

These picture descriptions as input will activate the subtitle SOM. Use the activated subtitle to retrieve the time stamp from the original subtitle f ile, and get the relevant movie screenshot based on it.





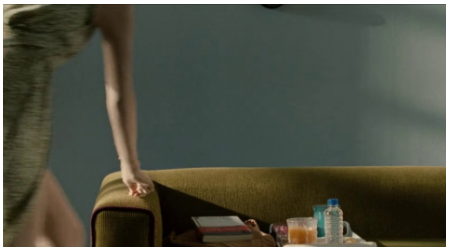
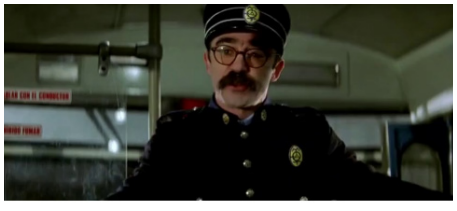








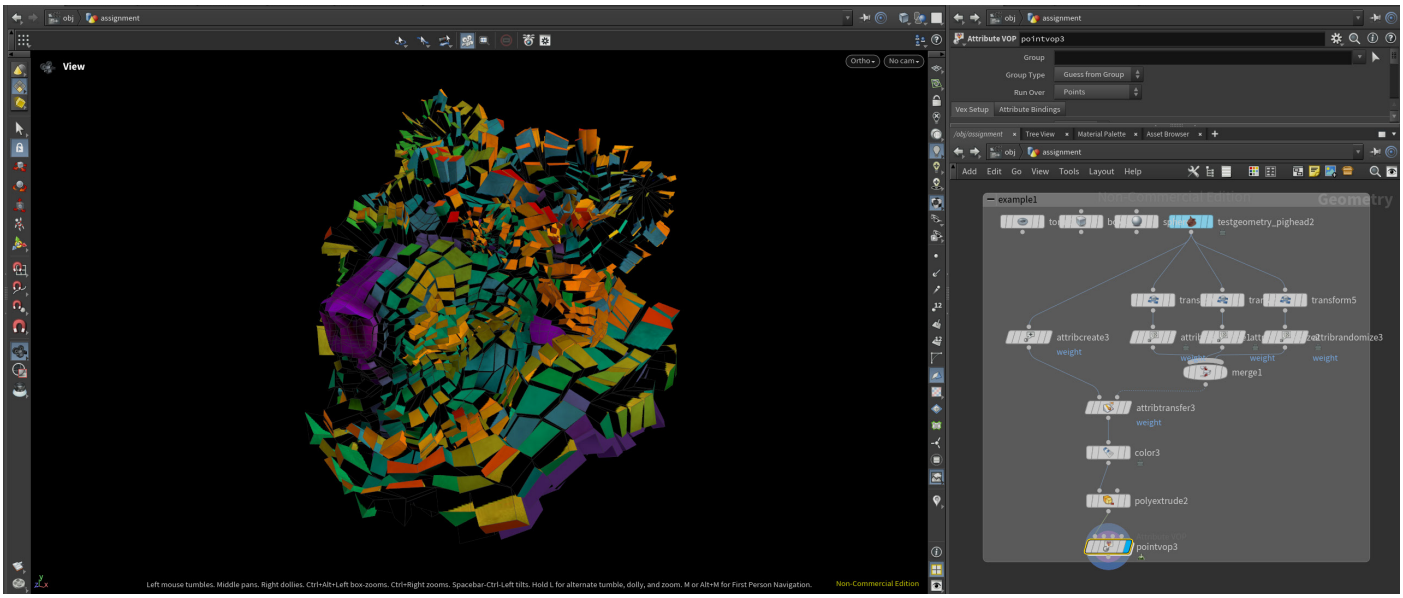


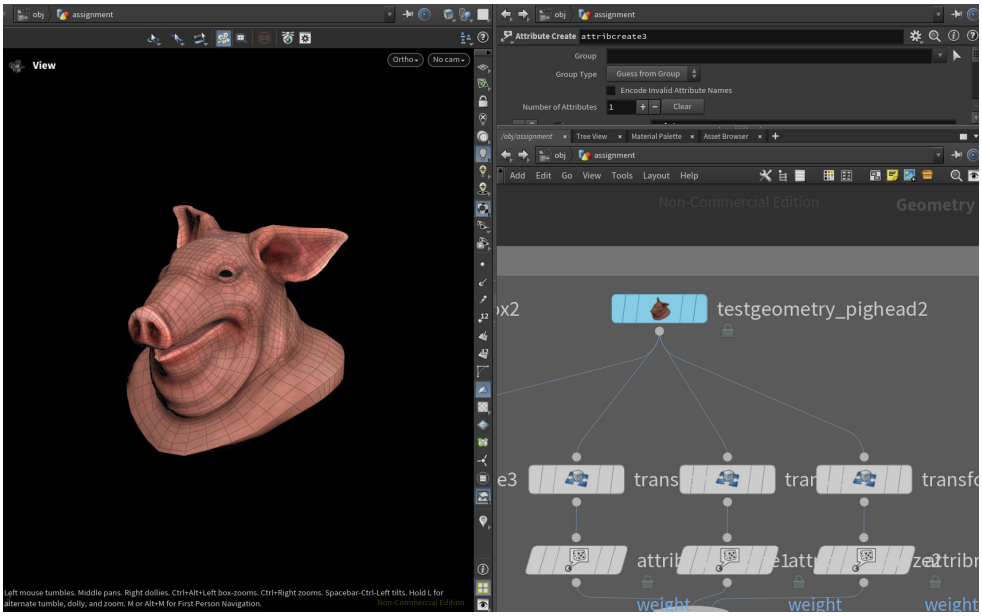




Github_Link: https://github.com/UD-Skills-2023-24/23121108/tree/main/Joris

RC11 Houdini Assignment
23121108

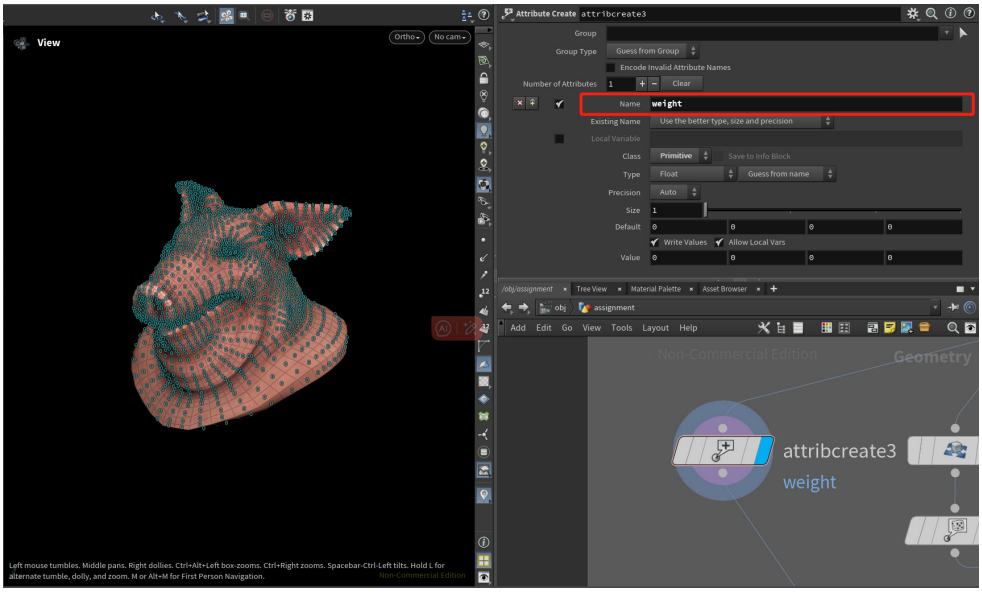
1{Houdini Fundamentals}
{Example1——Rainforest pig}





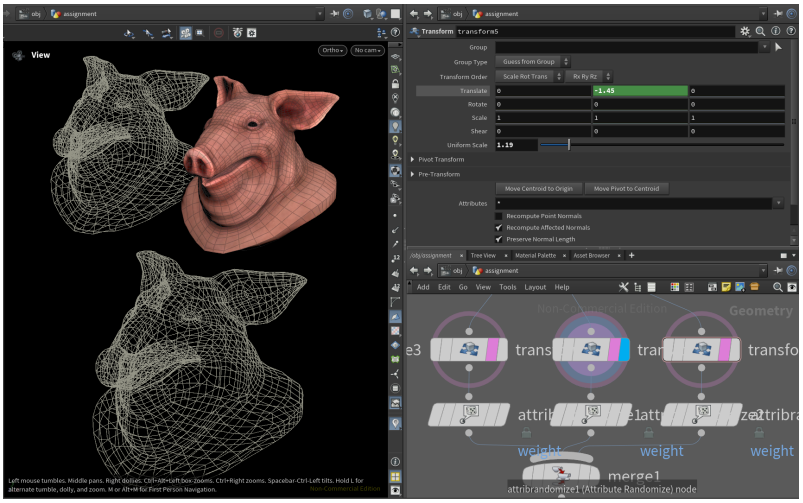
1Poghead SOP

This SOP is a geometry node that generates a Pighead shape. One of the geometric data types that contains points, primitives of type polygon, and vertices.



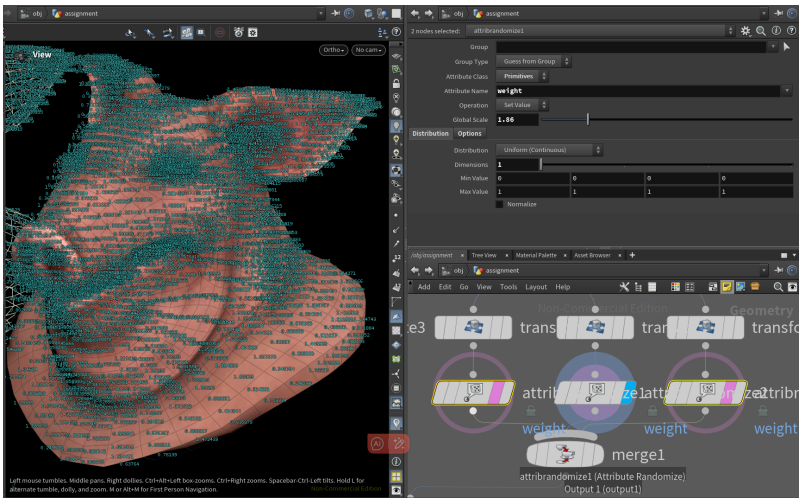
2AttribCreate SOP

This SOP is a node used to create or modify attributes on geometry. It adds the speciɒed attribute, "weight", to the geometry with the initial values "0.0".



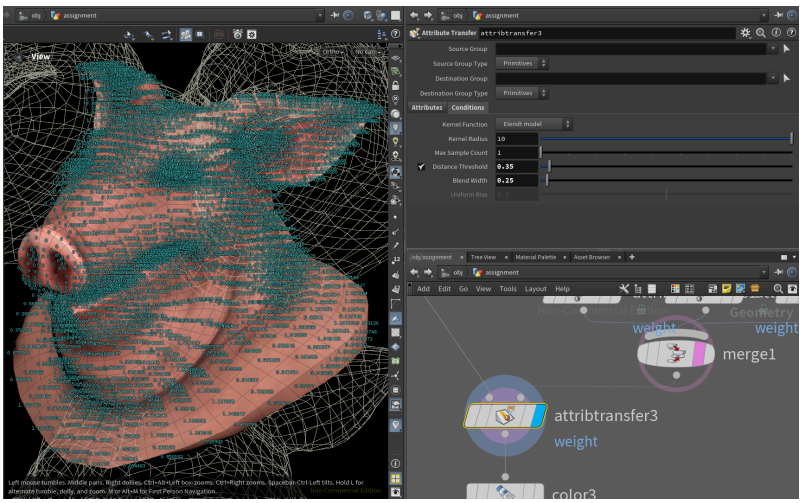
3Transform SOP

The Transform SOP in Houdini is a node used for translating, rotating, scaling, or generally transforming geometry in 3D space. Here I moved the pig's head in different directions and adjusted their size



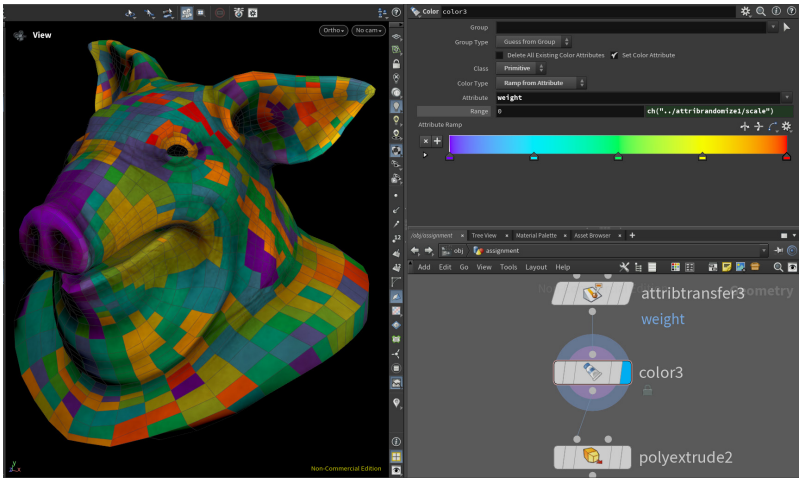
4Attribrandomize SOP

It is a node used to assign random values to attributes on geometry.I gave the attribute "weight" a new global numerical value.



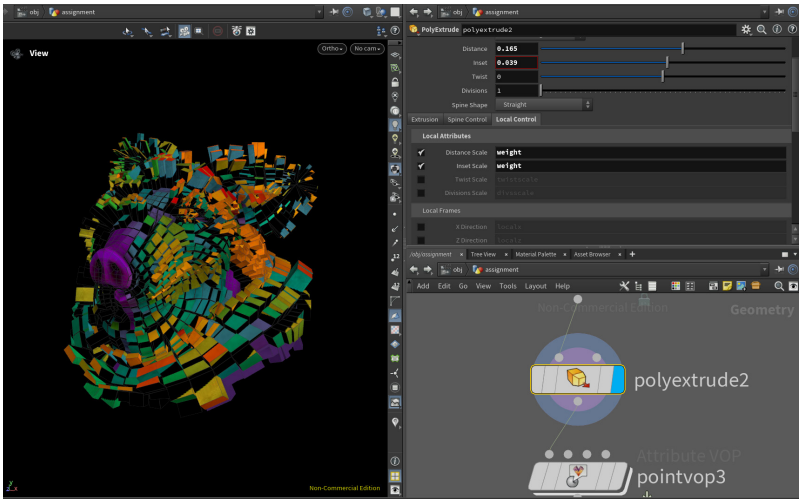
5AttribTransfer SOP

The "Attribute Transfer" SOP uses the "weight" primitive attribute to write out the distance value between the original sphere and the attractor. Here I'm passing random weights from the above three geometry into one geometry



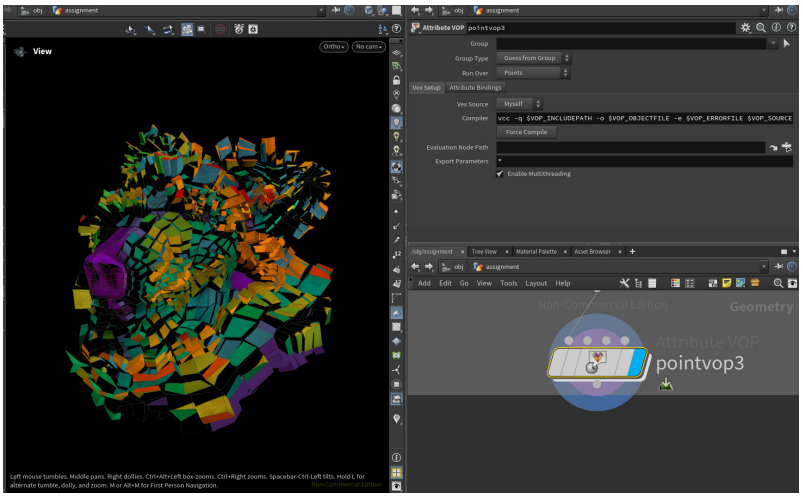
6Color SOP

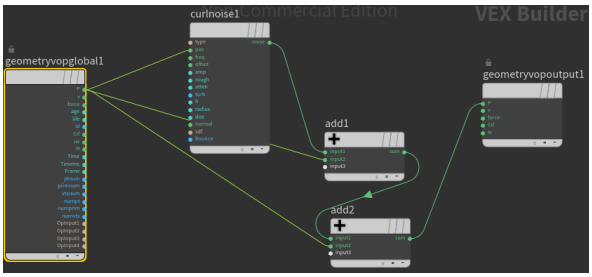
A visualisation of the "weight" primitive attribute according to the "Indra-Red" colorscheme.



7PolyExtrude SOP

We use the ”weight” value in ”distance scale” and use it as amultiplier of the extrusion value. Elements with higher "weight" values will be extruded further and with a larger inset.

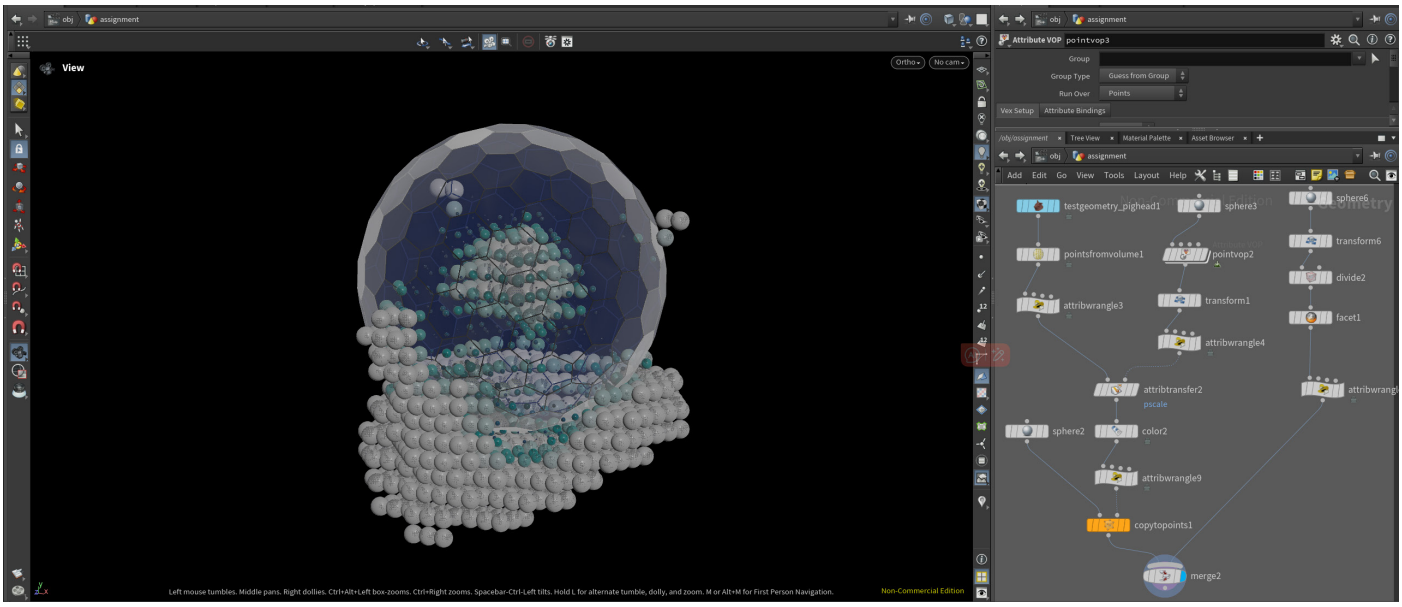


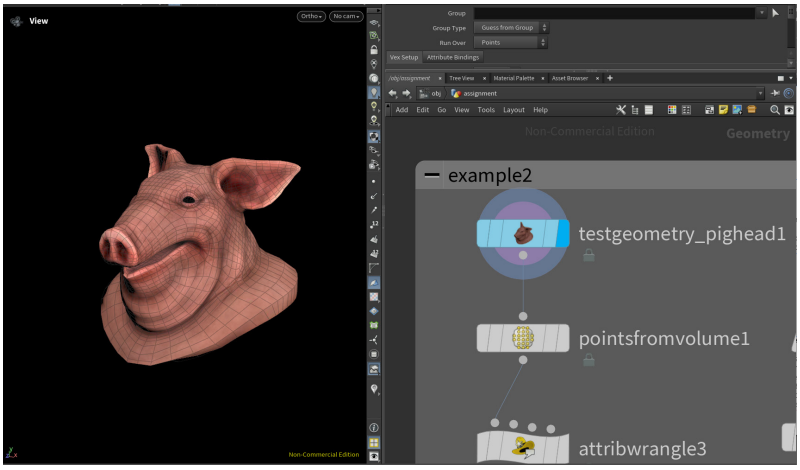


8Point VOP SOP

In this SOP I add some "noise" to inßuence the geometry. It is a visual programming interface.

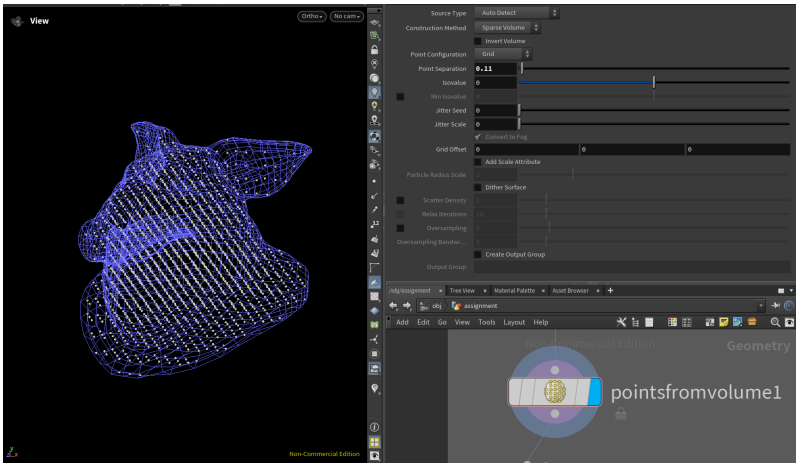
{Example2——Space pig}





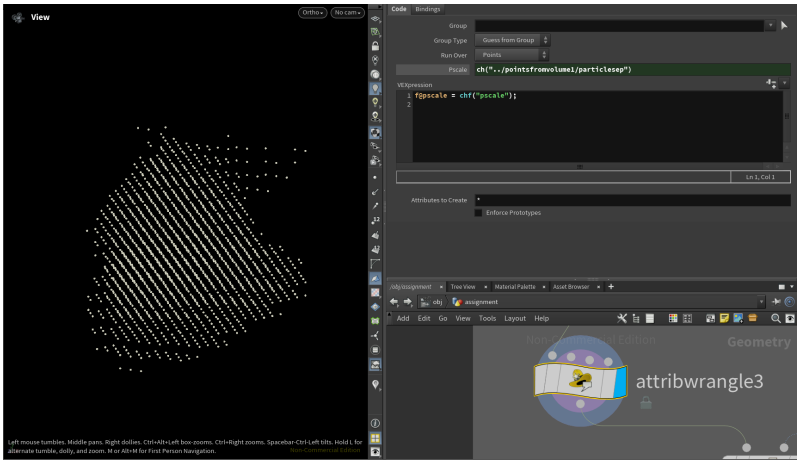
1Poghead SOP

This SOP is a geometry node that generates a Pighead shape. One of the geometric data types that contains points, primitives of type polygon, and vertices.



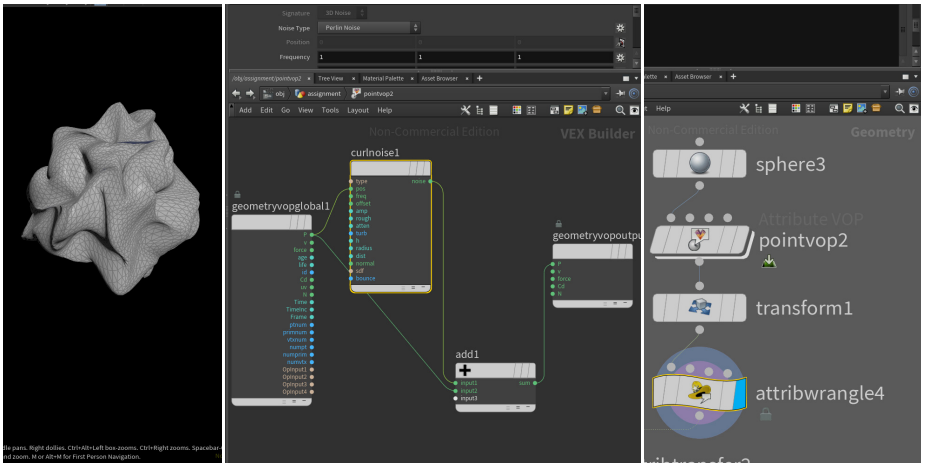
2PointsfromVolum SOP

Based on the volume of the pig's shape, I used this SOP to generate point clouds from volumetric data.



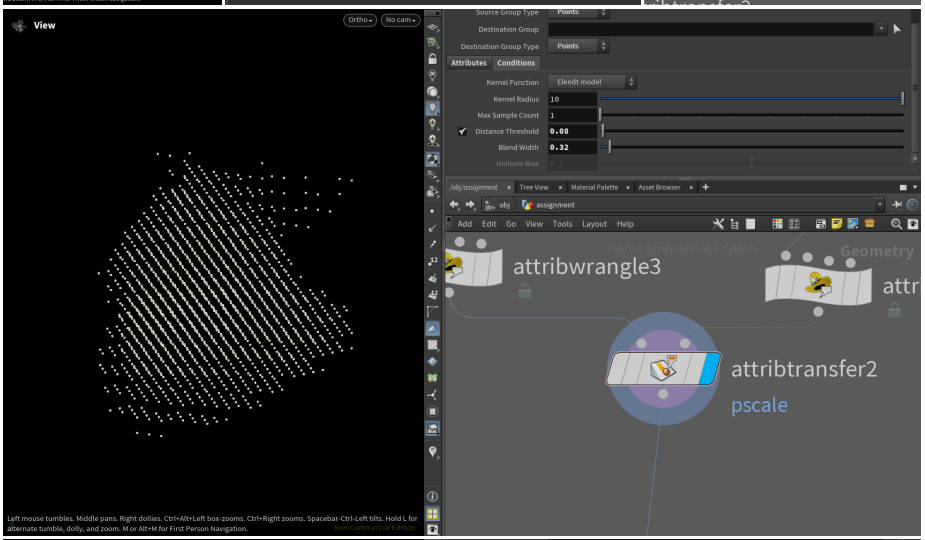
3AttribWrangle SOP

Writing code with VEX in this SOP allows me to assign the point size value entered on the pscale interface to the pscale attribute of the point currently being processed.



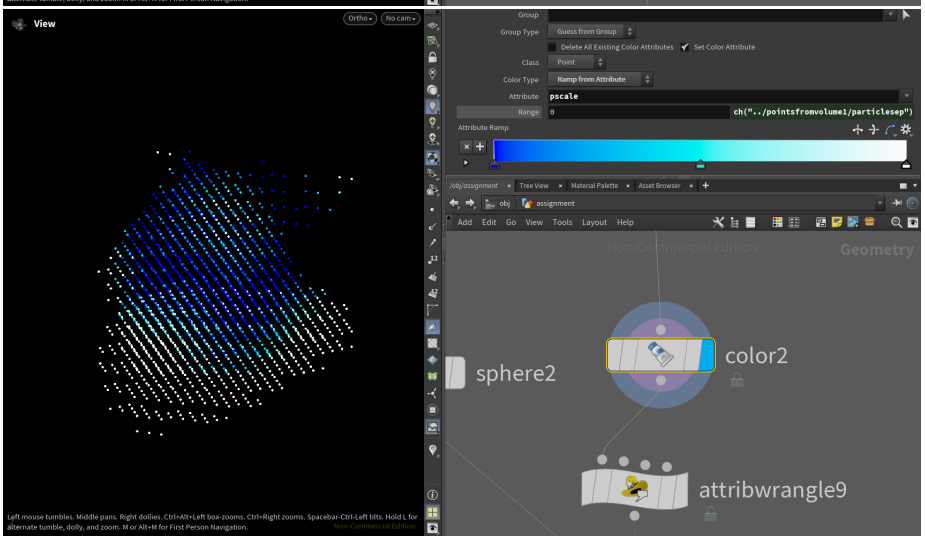
4Point VOP SOP

A sphere as input, was in fluenced by "noise" in pointvop . And next I used VEX (Vector Expression) language to set the "pscale" attribute of all points in the noised grid to 0.



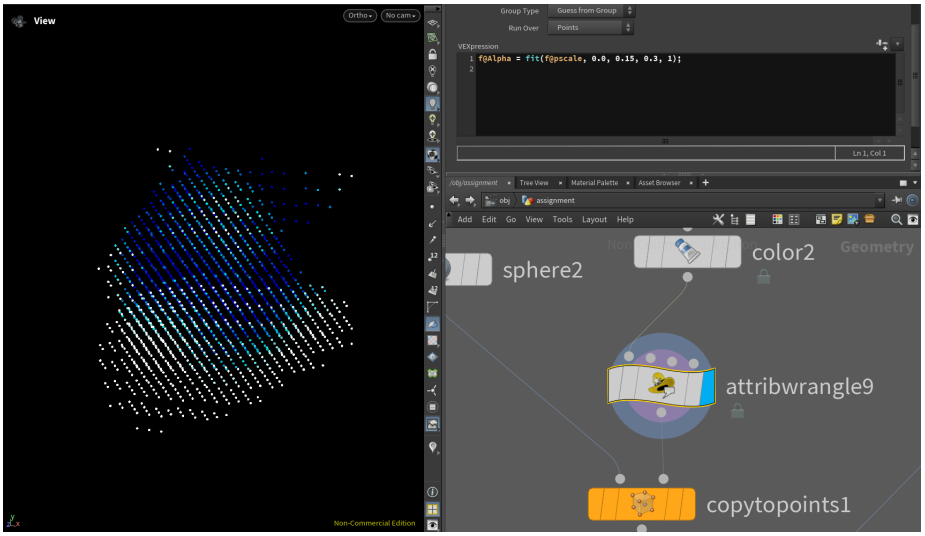
5AttribTransfer SOP

Use this SOP, I transfered the "pscale" attribute from the sphere to the pig. I also can control the scale of the pig based on the vale of "pscale" of the sphere geometry.



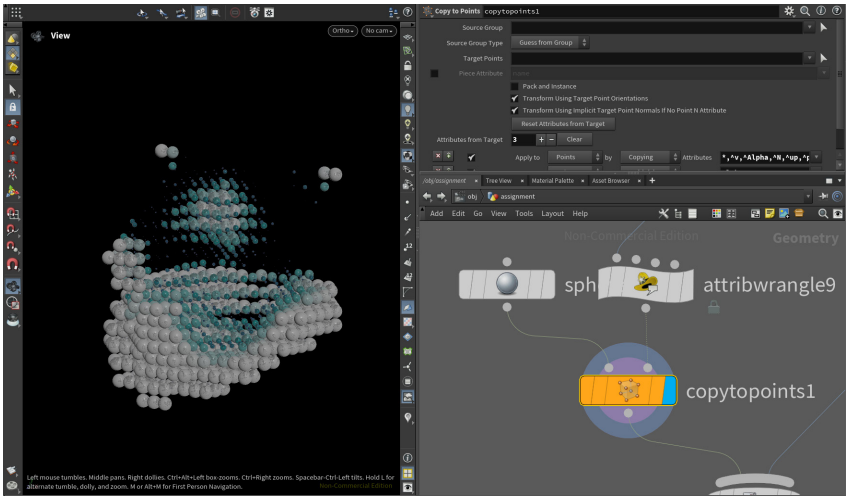
6Color SOP

Here I use "Whitewater" colorscheme to map the input values.



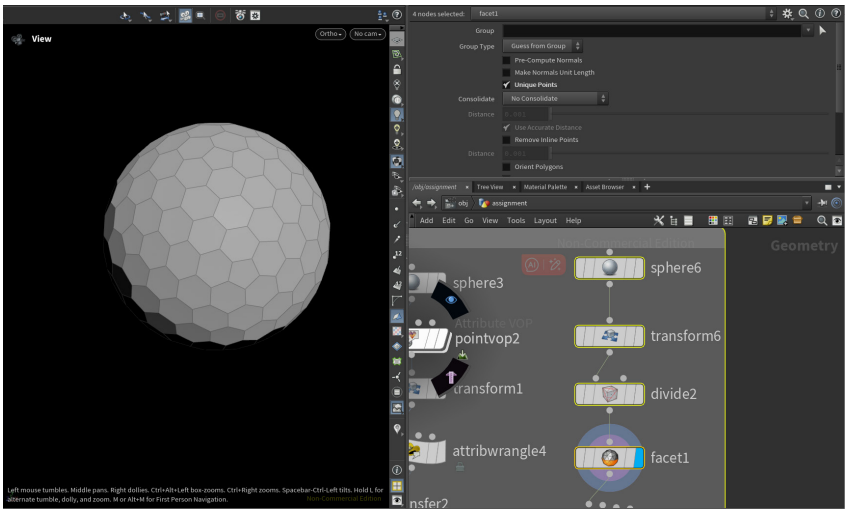
7AttribWrangle SOP

In this SOP, VEX expression is used to map the value of the pscale attribute of the current point from the interval [0.0, 0.15] to the interval [0.3, 1]. That is, set the Alpha property value based on the pscale property value of the point.



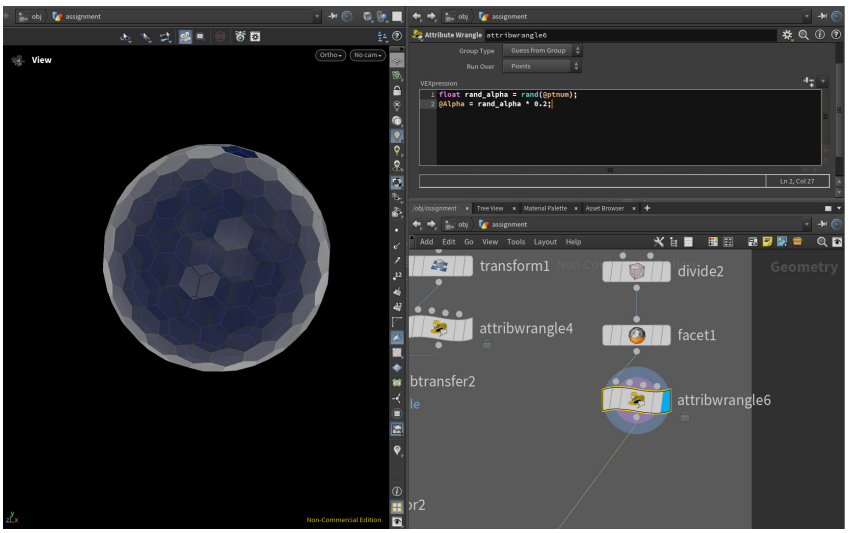
8CopytoPoints SOP

This node is is used to duplicate geometry onto the points of another piece of geometry. I placed copies of the spheres onto each point of the pig.



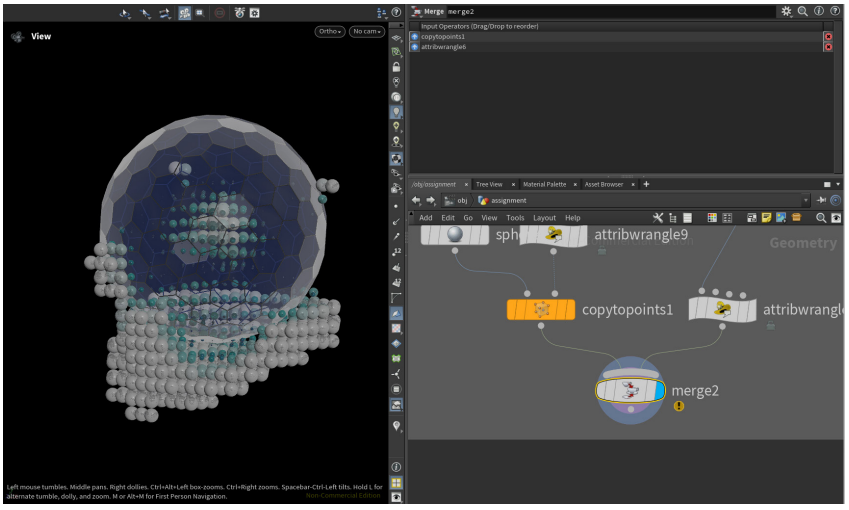
9Facet SOP

I input a Polygon sphere and subdivide it with "divide". Convert to a sheet using a "facet" vertices.



10AttribWrangle SOP

Here I use VEX code to generate a random number between 0 and 1. And make sure that each point has a unique random value. To make it even less transparent I multiply rand_alpha by 0.2 and assign the result to the point property Alpha. This scales the random number to a smaller range.point clouds from volumetric data.

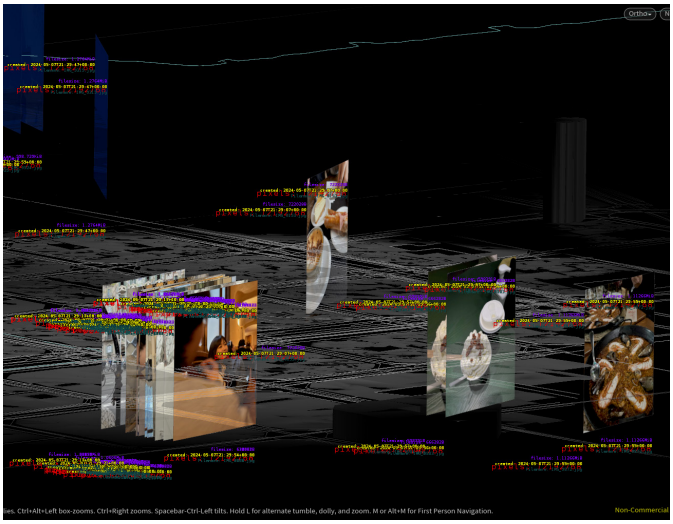


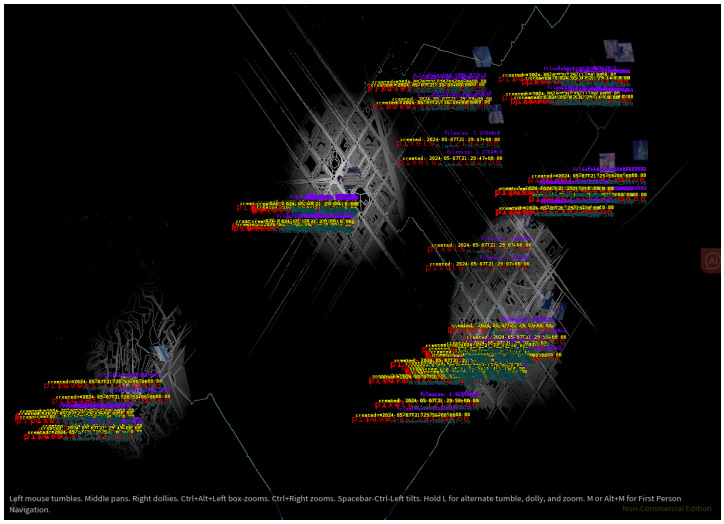
11Merge SOP

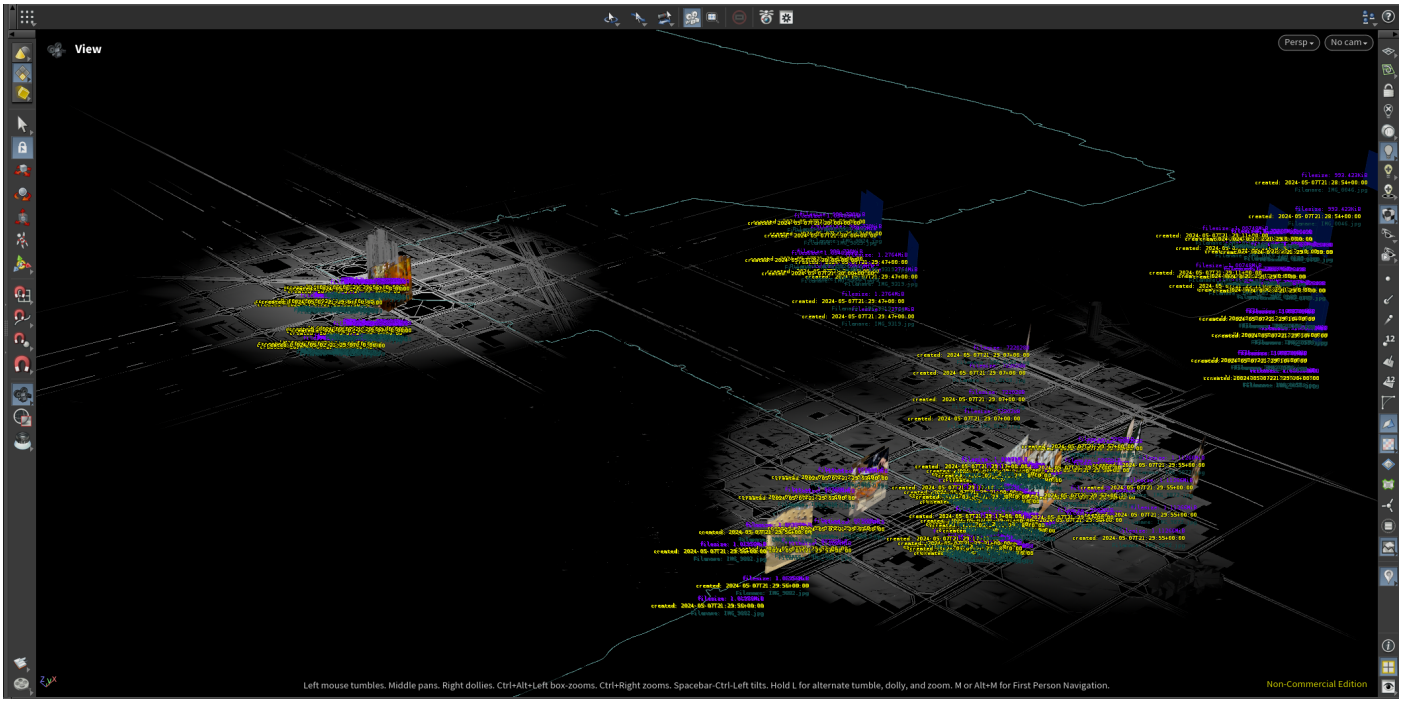
I use the merge SOP to merge the deformed pig's head and the transparent sphere into a single geometry. Make it easy to display simultaneously.

2{Visualizing GPS and Image Metadata}

In this workshop, I want to show my day in Barcelona. I import an image with geographic location information, extract the latitude and longitude and convert it to xy axis coordinates, from which I can determine the position of the image in the model. I also processed imported gmx and osm Þles in the same way. I converted the gmx Þle from Google Maps to gpx and imported it into houdini. I use it as a path for the camera to simulate my perspective. I also downloaded part of the osm Þle as a background for the city.

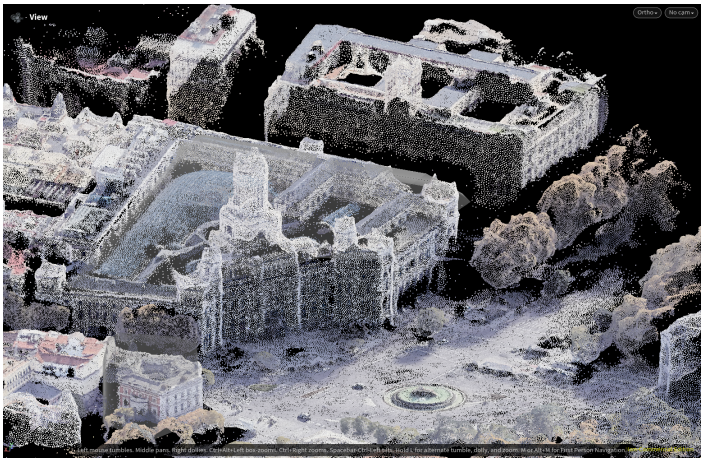


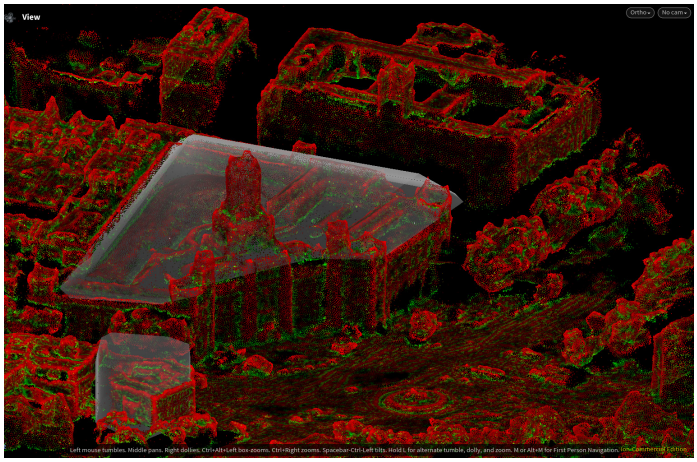


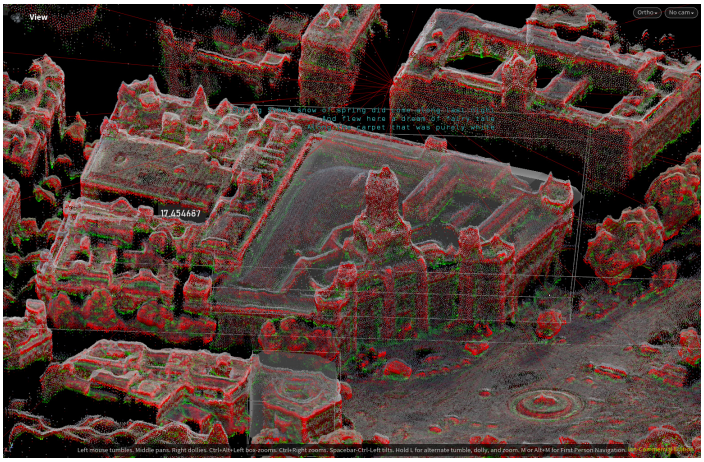


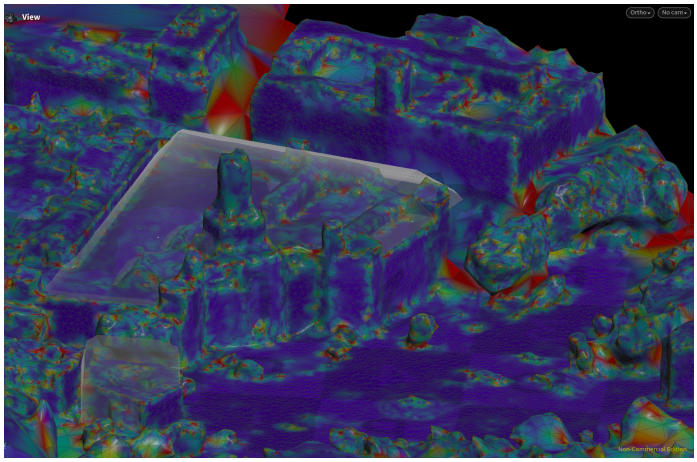
3{Video to 3D Model}

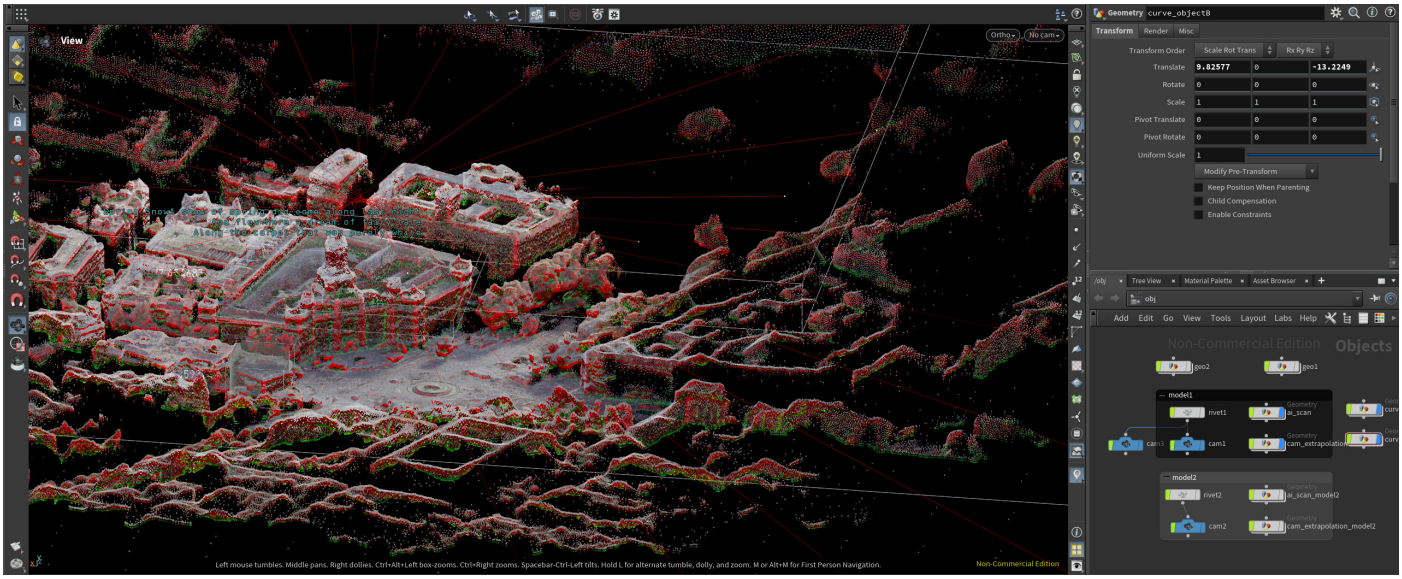
In this assignment, we chose the site of our project, the Cibeles Palace in Madrid, as the object of experiment. First, use google earth to get a video with a circular view. After processing the frame images with the pytho script, I used realitcapture to generate the mesh model. I isolated the camera information from the imported model and Reconstructed the camera path. For mesh model, we can convert it into point cloud. I subdivided the surface of the original mesh, calculated its curvature, and visualized it. I also used the volume to indirectly obtain the volume information of the building, and assigned it to the tag object to display it.







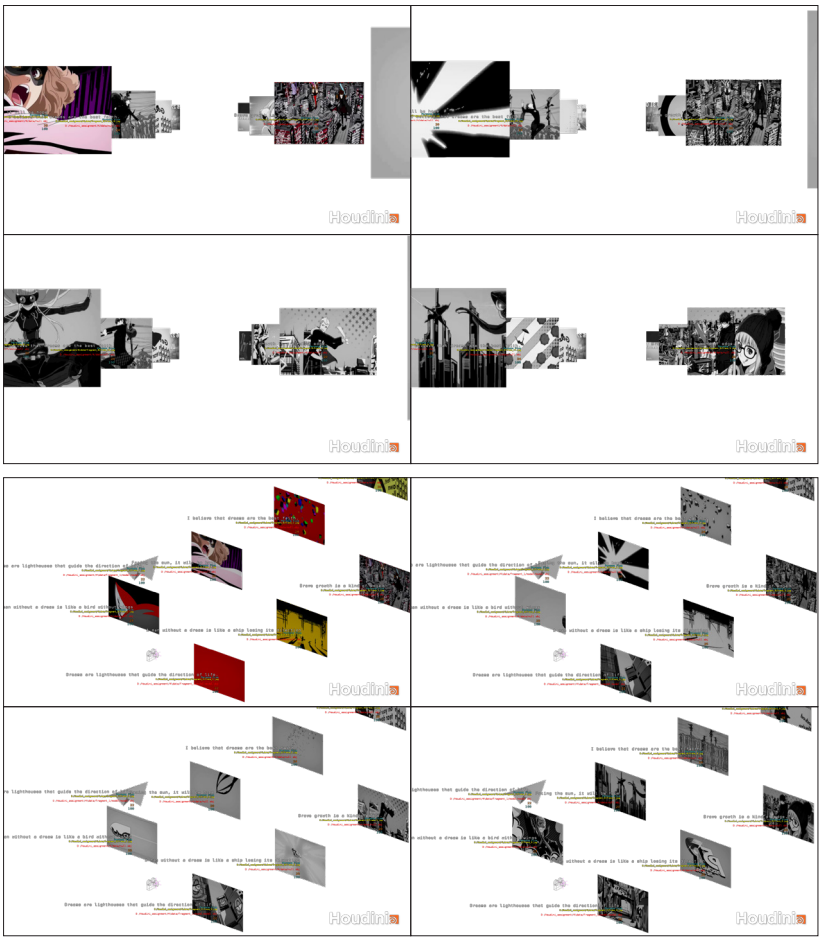


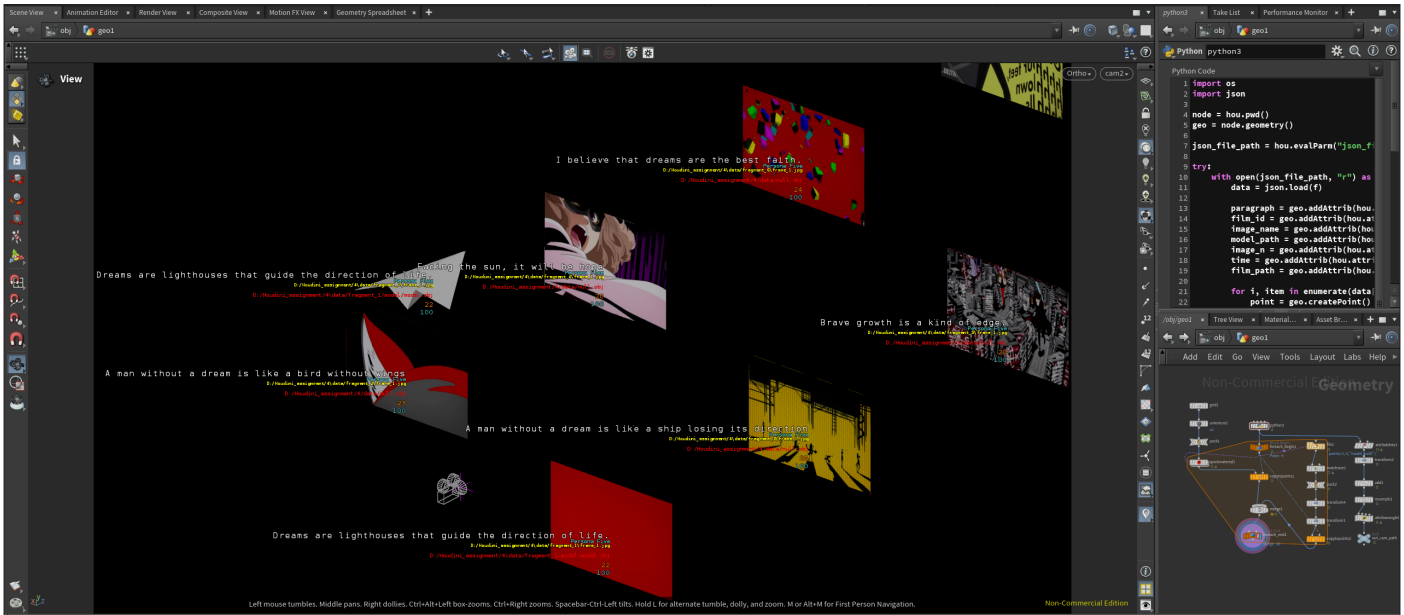


4{Visualizing JSON in Houdini}

In this workshop, I wrote a script to crawl some anime on youtubes. These videos are extracted into frames and trained into self-organizing maps. I broke a Tagore poem into a single sentence and activated SOM as input. I then store the resulting frames and sentences in a dictionary and save them as JSON Þles. Then I visualized them in Houdini. I also imported a model of a paper plane. Let it ßy up and down with the motion of the camera.

```
import json
# define
data = {
    "folder": [
        {
            "paragraph": "Dreams are lighthouses that guide the direction of life.",
            "filmID": "Persona Five",
            "image_name": "frame",
            "model_path": "model.obj",
            "image_n": 22,
            "time": 100,
            "film_path": "Steal your heart.mp4"
        },
        {
            "paragraph": "A man without a dream is like a bird without wings",
            "filmID": "Persona Five",
            "image_name": "frame",
            "model_path": "",
            "image_n": 23,
            "time": 100,
            "film_path": "Steal your heart.mp4"
        },
        {
            "paragraph": "A man without a dream is like a ship losing its direction",
            "filmID": "Persona Five",
            "image_name": "frame",
            "model_path": "",
            "image_n": 25,
            "time": 100,
            "film_path": "Steal your heart.mp4"
        },
        {
            "paragraph": "Facing the sun, it will be hope",
            "filmID": "Persona Five",
            "image_name": "frame",
            "model_path": "",
            "image_n": 20,
            "time": 100,
            "film_path": "Steal your heart.mp4"
        },
        {
            "paragraph": "Brave growth is a kind of edge.",
            "filmID": "Persona Five",
            "image_name": "frame",
            "model_path": "",
            "image_n": 20,
            "time": 100,
            "film_path": "Steal your heart.mp4"
        },
        {
            "paragraph": "I believe that dreams are the best faith.",
            "filmID": "Persona Five",
            "image_name": "frame",
            "model_path": "",
            "image_n": 24,
            "time": 100,
            "film_path": "Steal your heart.mp4"
        },
        {
```





Complexity Assignment - c.pierik

Github_Link: https://github.com/UD-Skills-2023-24/23121108/blob/main/Ceel/RC11_23121108_ASSIGNMENT.ipynb

Exercise One

1. Implement this algorithm in Python. Use the NumPy ndarray object foryour matrices;

In [16]:
```
import numpy as np
```

In [17]:
```
import numpy as np

def square_matrix_multiply(A, B):
    n = len(A)
    C = [[0] * n for _ in range(n)] #Initialize

    for i in range(n):# Iterate through rows of A
        for j in range(n):# Iterate through columns of B
            for k in range(n):# Iterate through elements of rows of A and columns of
                C[i][j] += A[i][k] * B[k][j]# Compute the product and accumulate it
    return C
```

In [3]:
```
# Example
A = np. array([[1,2],[3,4]])
B= np.array([[5,2],[7,8]])

result=square_matrix_multiply(A, B)
print(result)
```

```
[[19, 18], [43, 38]]
```

2. Give the asymptotic time complexity of the above algorithm or your implementation (they should be the same). Justify and explain your answer

I think the time complexity of O(n^3), where n is the size of the square matrices A and B.

"for i = 1 to n do"----------n times;

"for j = 1 to n do"----------n times;

"cij = 0 for k = 1 to n do"----------n times;

"cij = cij + aik × bkj"----------n times;

So, O(n *n* n) = O(n^3)

Suggestion1. Implement the algorithm with nested lists and compare the algorithms on different sizes of matrices

Suggestion2. Compare with built-in multiplication functions

In [18]:
```
import time

def generate_random_matrix(n):
    return [[np.random.randint(20) for _ in range(n)] for _ in range(n)]

# define the size
sizes = [100, 200, 500]
for size in sizes:
    A = generate_random_matrix(size)
    B = generate_random_matrix(size)

    # consuming time of built-in multiplication functions
    start_time = time.time()
    np_result = np.dot(A, B)
    numpy_time = time.time() - start_time

    # consuming time of nested list implementation
    start_time = time.time()
    list_result = square_matrix_multiply(A, B)
    list_time = time.time() - start_time

    print(f"Matrix size: {size}x{size}")
    print(f"NumPy time: {numpy_time:.3f} s")
    print(f"Nested list time: {list_time:.3f} s")
    print()
```

```
Matrix size: 100x100
NumPy time: 0.000 s
Nested list time: 0.143 s

Matrix size: 200x200
NumPy time: 0.005 s
Nested list time: 1.268 s

Matrix size: 500x500
NumPy time: 0.126 s
Nested list time: 27.045 s
```

Suggestion3. Look at multiplying more than two matrices, or at other operations on matrices

In [21]:
```
# let's try multiplying three matrices
def multiply_matrices(*matrices):
    result = matrices[0]
    for matrix in matrices[1:]:
        result = square_matrix_multiply(result, matrix)
    return result
```

In [19]:
```
# Example
A = generate_random_matrix(3)
B = generate_random_matrix(3)
C = generate_random_matrix(3)

print("random_matrix A:")
for row in A:
    print(row)
print("\nrandom_matrix B:")
for row in B:
```

```
    print(row)
print("\nrandom_matrix C:")
for row in C:
    print(row)
```

```
random_matrix A:
[1, 8, 12]
[10, 14, 10]
[16, 16, 12]

random_matrix B:
[19, 5, 18]
[0, 7, 14]
[17, 19, 19]

random_matrix C:
[19, 0, 17]
[3, 17, 15]
[4, 4, 11]
```

In [22]:
```
list_result = multiply_matrices(A, B, C)
print(list_result)

print()

np_result = np.dot(np.dot(A, B), C)
print(np_result)
```

```
[[6536, 6345, 12064], [10118, 8010, 17416], [13872, 10100, 23076]]

[[ 6536  6345 12064]
 [10118  8010 17416]
 [13872 10100 23076]]
```

Exercise Two

1. Describe and explain the algorithm. It should contain at least the following:

This algorithm, Square-Matrix-Multiply-Recursive (SMMRec), is a recursive implementation of matrix multiplication.

1. "SMMRec(A,B):": accepts two matrices A and B as input.
2. "n = nr of rows of A": Define a variable n which is equal to the number of rows of A, also the size of A.
3. "let C be a new n × n matrix": Create a new square C with the same dimensions as A and B.
4. "if n == 1 then":If the size of the matrix is 1 × 1, the following operations are performed; otherwise, the chunked multiplication of the matrix is performed.
5. "quarter matrices A, B, and C":Divide the matrices A, B and C into four equal-sized sub-matrices.
6. "C11 = SMMRec(A11,B11) + SMMRec(A12,B21)""C12 = SMMRec(A11,B12) + SMMRec(A12,B22)""C21 = SMMRec(A21,B11) + SMMRec(A22,B21)""C22 = SMMRec(A21,B12) + SMMRec(A22,B22)": Calculate the value of C11 C12 C21 C22 separately.

recursiveness: The base case of the recursion is when the size of the matrices becomes 1 × 1, the algorithm computes the product directly without further recursion.And in each recursion step, the algorithm divides the input matrices into four equal-sized submatrices. It then recursively multiplies these submatrices until the base case is reached, where it computes the product directly.

divide-and-conquer:

1. **Divide**:The algorithm divides the input matrices A, B, and C into four equal-sized submatrices.
2. **Conquer**: It recursively multiplies these submatrices. Each multiplication involves recursively calling the SMMRec function until the base case is reached, and then directly computing the product.
3. **Combine**: After computing the products of submatrices, the algorithm combines them to form the final result matrix C. The combination involves adding or summing the products of corresponding submatrices to form the corresponding submatrices of C.

2. Implement the recursive algorithm in Python. Reflect on which steps of the pseudocode were straightforward to implement and which hid a lot of complexity behind their language.

In [11]:
```python
import numpy as np
def split_matrix(A):
    mid = len(A) // 2

    A11 = [row[:mid] for row in A[:mid]]
    A21 = [row[:mid] for row in A[mid:]]
    A12 = [row[mid:] for row in A[:mid]]
    A22 = [row[mid:] for row in A[mid:]]

    return A11, A12, A21, A22

def combine_matrices(C11, C12, C21, C22):
    n = len(C11)
    C = [[0 for _ in range(2 * n)] for _ in range(2 * n)]

    for i in range(n):
        for j in range(n):
            C[i][j] = C11[i][j]
            C[i][j + n] = C12[i][j]
            C[i + n][j] = C21[i][j]
            C[i + n][j + n] = C22[i][j]
    return C

def matrix_add(A, B):
    return [[A[i][j] + B[i][j] for j in range(len(A))] for i in range(len(A))]




def SMM_recursive(A, B):
    n = len(A)
    if n == 1:
        # Base case
        return [[A[0][0] * B[0][0]]]
```

```
    else:

        A11, A12, A21, A22 = split_matrix(A)
        B11, B12, B21, B22 = split_matrix(B)

        # Recursive multiplications
        C11 = matrix_add(SMM_recursive(A11, B11), SMM_recursive(A12, B21))
        C12 = matrix_add(SMM_recursive(A11, B12), SMM_recursive(A12, B22))
        C21 = matrix_add(SMM_recursive(A21, B11), SMM_recursive(A22, B21))
        C22 = matrix_add(SMM_recursive(A21, B12), SMM_recursive(A22, B22))


        return combine_matrices(C11, C12, C21, C22)
```

In [19]:
```
# Example
def generate_random_matrix(n):
    return [[np.random.randint(20) for _ in range(n)] for _ in range(n)]

A = generate_random_matrix(4)
B = generate_random_matrix(4)
C = generate_random_matrix(4)

print("random_matrix A:")
for row in A:
    print(row)
print("\nrandom_matrix B:")
for row in B:
    print(row)
print("\nrandom_matrix C:")
for row in C:
    print(row)
```

```
random_matrix A:
[6, 8, 0, 13]
[16, 15, 2, 17]
[11, 9, 1, 17]
[19, 12, 1, 19]

random_matrix B:
[11, 15, 12, 19]
[15, 14, 8, 10]
[13, 2, 7, 14]
[15, 1, 0, 10]

random_matrix C:
[17, 7, 17, 7]
[1, 2, 1, 2]
[19, 9, 11, 15]
[14, 15, 1, 14]
```

In [22]:
```
C = SMM_recursive(A, B)
print("square matrix multiplyrecursive result:\n", C)

np_result = np.dot(A, B) # Check the results
print("np result:\n", np_result)
```

```
square matrix multiplyrecursive result:
 [[381, 215, 136, 324], [682, 471, 326, 652], [524, 310, 211, 483], [687, 474, 331,
685]]
np result:
 [[381 215 136 324]
 [682 471 326 652]
 [524 310 211 483]
 [687 474 331 685]]
```

Straightforward Steps:

1. Base case
2. Splitting matrices
3. Combining matrices

Steps with Complexity behind:

1. Matrix addition: Matrix addition involves adding corresponding elements of matrices together. We iterate over the matrices, accessing corresponding elements and adding them together in the matrix_add() function,
2. Recursive call: The algorithm computes the product of submatrices by performing addition and multiplication on the corresponding elements in the pseudocode.

3. Do a complexity analysis for the SMMRec algorithm. First comment on the complexity of the base case, divide step, conquer step, and combine step separately, then put it all together.

Use slides 20-21 of the lecture slides (pages 40-45 of the pdf) of Complexity lecture 2 as a guideline.

You do not have to do the tree analysis (slides 22-23), but do take aguess what the final complexity is.

Base Case: In the base case, when the matrices are of size 1x1, the complexity is constant time since there's only one element to process.

Divide Step: The divide step involves splitting the input matrices into smaller submatrices. This operation has a time complexity of O(1) since it only involves index manipulation.

Conquer Step: In the conquer step, the algorithm recursively computes the products of the submatrices. If we denote the size of the matrices as n, then in each recursive call, the matrices are divided into four submatrices of size n/2. Hence, the conquer step has a time complexity of T(n/2).

Combine Step: In the combine step, the products of the submatrices are combined to form the final result. Since this step involves summation and multiplication of corresponding elements, its time complexity is O(n^2), where n is the size of the matrices.

T(n) =

{

O(1) if n = 1 (base case)

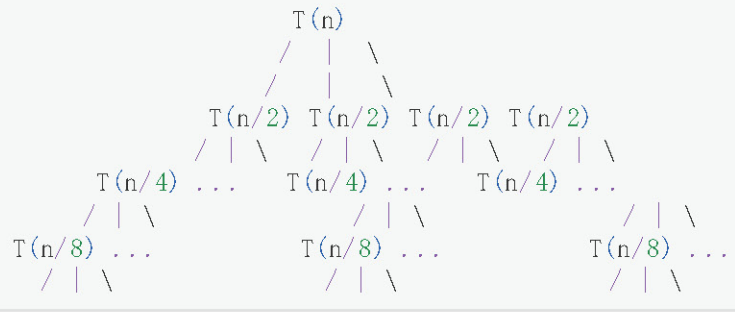
4T(n/2) + O(n^2) otherwise

}

The overall time complexity of the SMMRec algorithm is O(n^3), where n is the size of the input matrices.I guess.

Suggestion1. Do a tree analysis;

In [ ]:



In [ ]:

```
The number of nodes at each level of the tree:
Level 0: 1 node (root)
Level 1: 4 nodes
Level 2: 4^2 = 16 nodes
Level i: 4^i nodes
```

Since the tree height corresponds to the number of recursive calls until reaching the base case, and the base case occurs when n=1, the height of the tree is log2(n)

Therefore, the total number of nodes in the tree is 1+4+4^2+...+4^log2(n) The sum of this series is: 4n-1/3

Each node represents a subproblem, and the work done at each node (excluding the base case) is O(n^2) because of the combine step.Therefore, the total complexity of the algorithm is the product of the number of nodes and the work done at each node, which gives us:

T(n) = 4n-1/3 · O(n^2) = O(n^3)

Suggestion2. Test and compare the practical speed with the non-recursive algorithm

In [53]:

```
sizes = [4, 64, 256]
for size in sizes:
    A = generate_random_matrix(size)
    B = generate_random_matrix(size)

    # recursive algorithm
    start_time = time.time()
    result_recursive = SMM_recursive(A, B)
    recursive_time = time.time() - start_time
```

```
    # non-recursive algorithm
    start_time = time.time()
    result_non_recursive = np.dot(A, B)
    non_recursive_time = time.time() - start_time

    print(f"Matrix size: {size}x{size}")
    print(f"Recursive time: {recursive_time:.3f} s")
    print(f"Non-recursive time: {non_recursive_time:.3f} s")
    print()
```

```
Matrix size: 4x4
Recursive time: 0.002 s
Non-recursive time: 0.000 s

Matrix size: 64x64
Recursive time: 0.458 s
Non-recursive time: 0.000 s

Matrix size: 256x256
Recursive time: 37.168 s
Non-recursive time: 0.020 s
```

Exercise Three

1. Reflect on the difference between (complexity of) addition/subtraction and multiplication on matrices.

Addition/Subtraction Complexity: For addition and subtraction of matrices, the computational complexity is generally O（n^2）, where n is the size (number of rows or columns) of the matrices. This is because we simply iterate over each element of the matrices and perform addition or subtraction on corresponding elements.

Multiplication Complexity: Matrix multiplication has a higher computational complexity compared to addition/subtraction. The complexity arises from the fact that each element of the resulting matrix requires a sum of products of corresponding elements from the row of the first matrix and the column of the second matrix. The complexity of the standard matrix multiplication algorithm (like the naive approach or Strassen's algorithm) is approximately O(n^3).

2. Do a complexity analysis of the Strassen algorithm.

1. Base Case: The base case occurs when the matrices are small enough to be multiplied using a direct method, typically when the matrices reach a certain threshold size (often 2x2 or 3x3). The complexity of the base case is constant time, denoted as O(1).

2. Divide Step: The divide step involves partitioning the input matrices into submatrices. In the Strassen algorithm, each matrix is divided into four submatrices of approximately half the size. This step has a time complexity of O(1) since it only involves index manipulation.

3. Conquer Step The conquer step involves recursively multiplying the submatrices and combining the results to form the final result. In each recursive call, the algorithm performs 7

recursive multiplications of submatrices of size n/2 and 18 additions or subtractions of these intermediate results. Therefore, the complexity of the conquer step can be expressed as:

T(n) = 7T(n/2) + O(n2)

4. Combine Step: The combine step involves combining the results of the recursive calls to form the final result matrix. This step typically involves simple additions and subtractions of corresponding submatrices, which has a time complexity of O(n^2)

T(n) =

{

O(1) if n is small enough (base case)

7T(n/2) + O(n^2) ≈ O(n^2.81) otherwise

}

Suggestion1. Implement and test the algorithm;

In [56]:
```
def matrix_sub(A, B):
    return [[A[i][j] - B[i][j] for j in range(len(A))] for i in range(len(A))]

def Strassen(A, B):
    n = len(A)
    if n == 1:
        # Base case
        return [[A[0][0] * B[0][0]]]
    else:
        # Divide
        A11, A12, A21, A22 = split_matrix(A)
        B11, B12, B21, B22 = split_matrix(B)

        S1 = matrix_sub(B12, B22)
        S2 = matrix_add(A11, A12)
        S3 = matrix_add(A21, A22)
        S4 = matrix_sub(B21, B11)
        S5 = matrix_add(A11, A22)
        S6 = matrix_add(B11, B22)
        S7 = matrix_sub(A12, A22)
        S8 = matrix_add(B21, B22)
        S9 = matrix_sub(A11, A21)
        S10 = matrix_add(B11, B12)

        # Conquer
        P1 = Strassen(A11, S1)
        P2 = Strassen(S2, B22)
        P3 = Strassen(S3, B11)
        P4 = Strassen(A22, S4)
        P5 = Strassen(S5, S6)
        P6 = Strassen(S7, S8)
        P7 = Strassen(S9, S10)

        C11 = matrix_add(matrix_sub(matrix_add(P5, P4), P2), P6)
        C12 = matrix_add(P1, P2)
        C21 = matrix_add(P3, P4)
        C22 = matrix_sub(matrix_sub(matrix_add(P5, P1), P3), P7)
```

```
        # Combine
        return combine_matrices(C11, C12, C21, C22)
```

In [61]:
```
# Example
A = [[1, 2, 3, 12], [4, 5, 6, 2], [9, 7, 10, 12], [11, 10, 5, 7]]
B = [[2, 14, 8, 5], [8, 17, 2, 2], [16, 4, 10, 12], [12, 7, 7, 7]]

C = Strassen(A, B)
print("Strassen result:")
for row in C:
    print(row)
print()

np_result = np.dot(A, B)
print("np result:\n", np_result)
```

```
Strassen result:
[210, 144, 126, 129]
[168, 179, 116, 116]
[378, 369, 270, 263]
[266, 393, 207, 184]

np result:
 [[210 144 126 129]
 [168 179 116 116]
 [378 369 270 263]
 [266 393 207 184]]
```

In [62]:
```
# Time comparison
sizes = [4,64,256]
for size in sizes:
    A = generate_random_matrix(size)
    B = generate_random_matrix(size)

    # Strassen's algorithm
    start_time = time.time()
    result_strassen = Strassen(A, B)
    strassen_time = time.time() - start_time

    # Recursive algorithm
    start_time = time.time()
    result_recursive = SMMRec(A, B)
    recursive_time = time.time() - start_time

    # Non-recursive algorithm
    start_time = time.time()
    result_non_recursive = np.dot(A, B)
    non_recursive_time = time.time() - start_time


    print(f"Matrix size: {size}x{size}")
    print(f"Strassen's time: {strassen_time:.3f} s")
    print(f"Recursive time: {recursive_time:.3f} s")
    print(f"Non-recursive time: {non_recursive_time:.3f} s")
```

```
Matrix size: 4x4
Strassen's time: 0.001 s
Recursive time: 0.001 s
Non-recursive time: 0.000 s
Matrix size: 64x64
Strassen's time: 0.705 s
Recursive time: 0.577 s
Non-recursive time: 0.000 s
Matrix size: 256x256
Strassen's time: 34.941 s
Recursive time: 37.086 s
Non-recursive time: 0.020 s
```

Suggestion2. Discuss other optimisations;

Finkel-Kanth algorithm

FK algorithm (Finkel-Kanth algorithm) is an optimization algorithm for sparse matrix multiplication, aiming to reduce the number of multiplication operations by exploiting the sparsity property. Sparse matrices are matrices where the majority of elements are zeros.The key idea of the FK algorithm is to only compute multiplication operations for non-zero elements in sparse matrices, while ignoring zero elements. By doing so, the computational workload can be significantly reduced, leading to improved efficiency in sparse matrix multiplication.

The FK algorithm typically consists of two stages: preprocessing stage and computation stage.

1 Preprocessing Stage: In the preprocessing stage, the input sparse matrix undergoes some preprocessing operations to determine which elements are non-zero and their positions. This can be achieved by scanning the rows and columns of the sparse matrix and storing the positions of non-zero elements in a data structure, such as compressed sparse row (CSR) or compressed sparse column (CSC) representation.

2 Computation Stage: In the computation stage, utilizing the positions of non-zero elements obtained from the preprocessing stage, multiplication operations are only performed for these non-zero elements. This means that for two sparse matrices A and B, multiplication operations are only performed when there are non-zero elements in the same position in row i of matrix A and column j of matrix B, and the results are accumulated at the corresponding position in the product matrix.

Here is an example that explains the two stages

In [65]:
```python
def preprocess_sparse_matrix(matrix):
    non_zero_positions = []
    for i in range(len(matrix)):
        for j in range(len(matrix[0])):
            if matrix[i][j] != 0:
                non_zero_positions.append((i, j))
    return non_zero_positions

def finkel_kanth_multiply(A, B):
    # Preprocessing stage
    non_zero_positions_A = preprocess_sparse_matrix(A)
    non_zero_positions_B = preprocess_sparse_matrix(B)
```

```
    # Computation stage
    result = [[0 for _ in range(len(B[0]))] for _ in range(len(A))]
    for i, j in non_zero_positions_A:
        for k, l in non_zero_positions_B:
            if j == k:  # Only perform multiplication when the column index of A mat
                result[i][l] += A[i][j] * B[k][l]
    return result

# Example
A = [[1, 0, 0], [0, 2, 0], [0, 0, 3]]
B = [[4, 0, 0], [0, 5, 0], [0, 0, 6]]

print("Preprocessing stage:")
print("Non-zero positions in matrix A:", preprocess_sparse_matrix(A))
print("Non-zero positions in matrix B:", preprocess_sparse_matrix(B))
print()

print("Computation stage:")
result = finkel_kanth_multiply(A, B)
for row in result:
    print(row)
```

```
Preprocessing stage:
Non-zero positions in matrix A: [(0, 0), (1, 1), (2, 2)]
Non-zero positions in matrix B: [(0, 0), (1, 1), (2, 2)]

Computation stage:
[4, 0, 0]
[0, 10, 0]
[0, 0, 18]
```

Advantages: Reduction in the number of multiplication operations, especially for sparse matrices, leading to significant computational savings.Improved efficiency by ignoring zero elements, particularly beneficial for large sparse matrices.

Limitations: The preprocessing stage may require additional time and space to determine the positions of non-zero elements in sparse matrices, especially for extremely sparse matrices.The FK algorithm is suitable for sparse matrices and may not perform as well as traditional matrix multiplication algorithms for dense matrices.