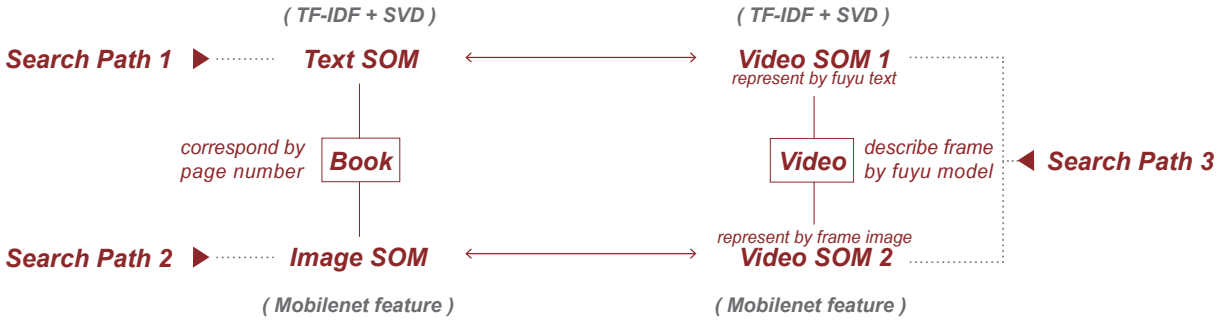
FINAL_ASSIGNMENT

RC11_23114486

FINAL_ASSIGNMENT_1
EXPLANATION AND OUTCOMES

Tutor: Julian Besems

For this assignment, I built this 'search engine' using some materials in the history textbook. There are 3 datasets and 3 different domains. Their vectorization and correlation are as follows:



Search Path 1: Input Text to search

1. ***Input text to Text SOM to find the most similar paragraph and correspongding images by page number (Results 1 and 2);***
2. ***Input the result text into Video SOM1 as a query, to find the film through the most matching fuyu description (Result 3);***
3. ***Or input the images into Video SOM2 as a query to find the flim by image similarity (Result 4).***

Search Path 2: Input Image to search

1. ***Input image to Image SOM and search for the most similar picture and the text in the same page (Results 1 and 2);***
2. ***Input the result image into Video SOM2 as a query, and the film is found by the image similarity (Result 3);***
3. ***Or input the corresponding text into Video SOM1 to find the film by the most matching fuyu description (Result 4).***

Search Path 3: Input Video to search

1. ***Input a video, extract a certain frame as an image, and input it into Video SOM2 to find the most similar flim (Result 1);***
2. ***Input the cover of the result film into Image SOM, get the most similar picture and corresponding text (Results 2 and 3).***
3. ***Or input fuyu description of the result film as query into Text SOM to find the most matching sentence (Result 4);***

Specific explanations and outcomes are as follows. The explanation of the required code is placed later in the whole process:
'Contrast Vectorization Method' is in 2.1.2 and 2.2.2,
'Plot QE and TE' and 'PCA' are located in 2.1.3,
'Use separate .py file with class' is located in 3.

1. Build dataset

Considering the particularity of the textbook, the text and the corresponding picture have a strong connection, so when extracting the text and picture from the PDF, it is corresponding to the page number and saved as a dictionary to ensure that one type of data can be found according to another type of data.

Finally get a .json file:


```
[
  {
    "book_id": "World_History_Volume_1",
    "pages": [
      {
        "page_id": "page_19",
        "texts": [
          "Philip Eckebrecht for the noted German astronomer Johannes Kepler, gives a sense of the breadth of territory this text will cover. As we see later in this chapter, maps often reflect the maker's perception of geographical realities. (credit: modification of work "A Modern Depiction of the World" by Library of Congress/Wikimedia Commons, Public Domain)",
          "Angelou suggested—a way to meet the pain of the past and overcome it? Or is it, as Winston Churchill said, a chronicle by the victors, an interpretation by those who write it? History is all this and more. Above all else, it is a path to knowing why we are the way we are—all our greatness, all our faults—and therefore a means for us to understand ourselves and change for the better.",
          "But history serves this function only if it is a true reflection of the past. It cannot be a way to mask the darker parts of human nature, nor a way to justify acts of previous generations. It is the historian's task to paint as clear a picture as sources will allow.",
          "Will history ever be a perfect telling of the human tale? No. There are voices we may never hear. Yet each new history book written and each new source uncovered reveal an ever more precise record of events around the world (Figure 1.1). You are about to take a journey into human history."
        ],
        "images": [
          "/content/drive/MyDrive/main_design/textbook/history/image_world_history_1/page_19_image_0.png"
        ]
      },
      {
        "page_id": "page_20",
        "texts": [
          "From the legends of Troy heralded by Homer to the contents of digital archives accessed by modern students, the human story has fascinated and instructed those who have tried to understand its complexities. Knowing the past has long been considered a mark of civilization, and its study has never been more important. We have all heard the philosopher George Santayana's observation, "Those who do not learn from history are doomed to repeat it." Yet because history is an ever-changing collection of events influenced and shaped by a variety of causes and outcomes, it never truly repeats at all. Santayana's comment rings true, however, in that we can discern patterns of
```


2. Train SOM and Create a search function

2.1 Train Text SOM

Code: *https://github.com/UD-Skills-2023-24/github-codebook-RC11-23114486/blob/main/Final_Assignment_1_Julian/Text_SOM.ipynb*

2.1.1 Preprocess and Vectorize

Load text from json file and create a new dictionary to ensure that the corresponding images could be found by sentence.

```
[ ] with open('/content/drive/MyDrive/main_design/textbook/history/text_with_image_combinebook.json','r',encoding='utf-8')as file:
        history = json.load(file)
```

```
[ ] history[0]["pages"]
```

```
[ ] cleaned_history = []
    for book in history:
        cleaned_book = {'book_id': book['book_id'],'pages':[]}
        for page in book['pages']:
                cleaned_texts = [re.sub(r'\(.*?\)', '', paragraph)for paragraph in page['texts']]
                cleaned_page = {
                        "page_id": page["page_id"],
                        "texts": cleaned_texts,
                        "images": page["images"]
                        }
                cleaned_book['pages'].append(cleaned_page)
        cleaned_history.append(cleaned_book)
```

```
[ ] sentence_find_page = {}
    for book in cleaned_history:
        book_id = book['book_id']
        for page in book['pages']:
            page_id = page['page_id']
            for sentence in page['texts']:
                sentence_find_page[sentence] = (book_id, page_id, page['images'])
```

```
[ ] test="Philip Eckebrecht for the noted German astronomer Johannes Kepler, gives a sense of the breadth of territory this text will cover. As we see
    book_id, page_id, images = sentence_find_page.get(test, (None, None, None))
    print(images)
```

```
['/content/drive/MyDrive/main_design/textbook/history/image_world_history_1/page_19_image_0.png']
```

Try three vectorization methods: TF-IDF, TF-IDF+SVD, and Doc2Vec.

For the same query sentence, calculate the cosine similarity to match the most similar sentences, check the top5 results.

For example, test_1 results in the following:

query_words = "Science and technology have developed in this century."

TF-IDF

Index: 141, Similarity Score: 0.24282283420327977
original_texts[141]:

"have looked like demonstrates a striking resemblance to modern humans. " by Public Library of Science/Wikimedia Commons, CC BY 2.5)

Index: 292, Similarity Score: 0.22995127576599567
original_texts[292]:

The fourth millennium BCE in Sumer was also a period of technological innovation. One important invention made after 4000 BCE was the process for manufacturing bronze, an alloy of tin and copper, which marked the beginning of theBronze Agein Mesopotamia. In this period, bronze replaced stone as the premier material for tools and weapons and remained so for nearly three thousand years. The ancient Sumerians also developed the plow, the wheel, and irrigation techniques that used small channels and canals with dikes for diverting river water into fields. All these developments allowed for population growth and the continued rise of cities by expanding agricultural production and the distribution of agricultural goods. In the area of science, the Sumerians developed a sophisticated mathematical system based on the numbers sixty, ten, and one.

TF-IDF SVD

Index: 128, Similarity Score: 0.4377719369971416
original_texts[128]:

The concept of evolution over time is one we are all likely familiar with. Consider, for example, how technology has evolved. The first true smartphones appeared on the market at the beginning of this century, but these complicated devices didn't spring all at once from the minds of ambitious engineers. Rather, these engineers built on technology that had evolved and improved over many decades. In the mid-1800s, telegraph technology first demonstrated that electricity could be used for long-distance communication. That technology paved the way for the first telephones, which were basic and expensive but over many decades became more sophisticated, more common, and cheaper. By the early 1980s, electronics companies had begun selling telephones that used radio technology to communicate wirelessly. Over time these devices were made faster and smaller, and companies added features like cameras, microprocessors, and eventually internet access. With these evolutionary transformations, the smartphone was born.

Index: 891, Similarity Score: 0.3745981721426376
original_texts[891]:

These wars and invasions coincided with an important technological innovation, the birth of sophisticated iron-making technology. For thousands of years, bronze had been the metal of choice in the ancient world. But the disruptions caused by the Late Bronze Age Collapse made it difficult for metal workers to access tin, a crucial ingredient in bronze. Without a sufficient supply of tin, artisans experimented for centuries with iron ore. In the process, they developed the techniques of steeling , quenching , and tempering to produce a metal far superior in strength to bronze. By around 900 BCE, theIron Agehad begun in the eastern Mediterranean.

Doc2Vec

Index: 128, Similarity Score: 0.43767309188842773

Index: 500, Similarity Score: 0.4260444939136505
original_texts[500]:

• What set the military of the Neo-Assyrians apart from their rivals? How did their use of technology increase the severity and frequency of warfare in the Near East?

TF-IDF measures the uniqueness of words in a text, tending to focus on the frequency and significance of the words in the query sentence relative to the words in the documents.
After dimensionality reduction with SVD, the main features of the original data were captured, and some noise was removed.
Doc2Vec captures words within a certain range before and after a word, focusing more on understanding the contextual relationships within documents.

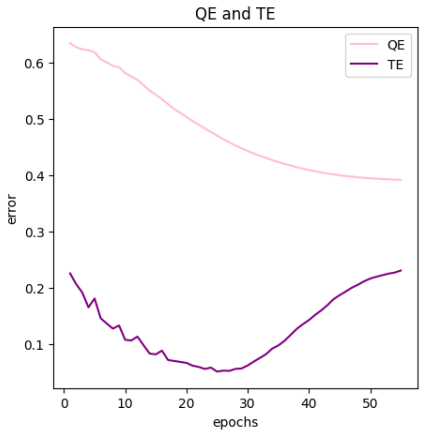
Based on the test results of 5 different query sentences, TF-IDF+SVD offers an understanding of the dataset closest to what was expected. So chose to use this vectorization method.

2.1.2 Train SOM

Modify the train SOM function, calculate the current QE and TE after each epochs, saving the SOM, and QE and TE values.

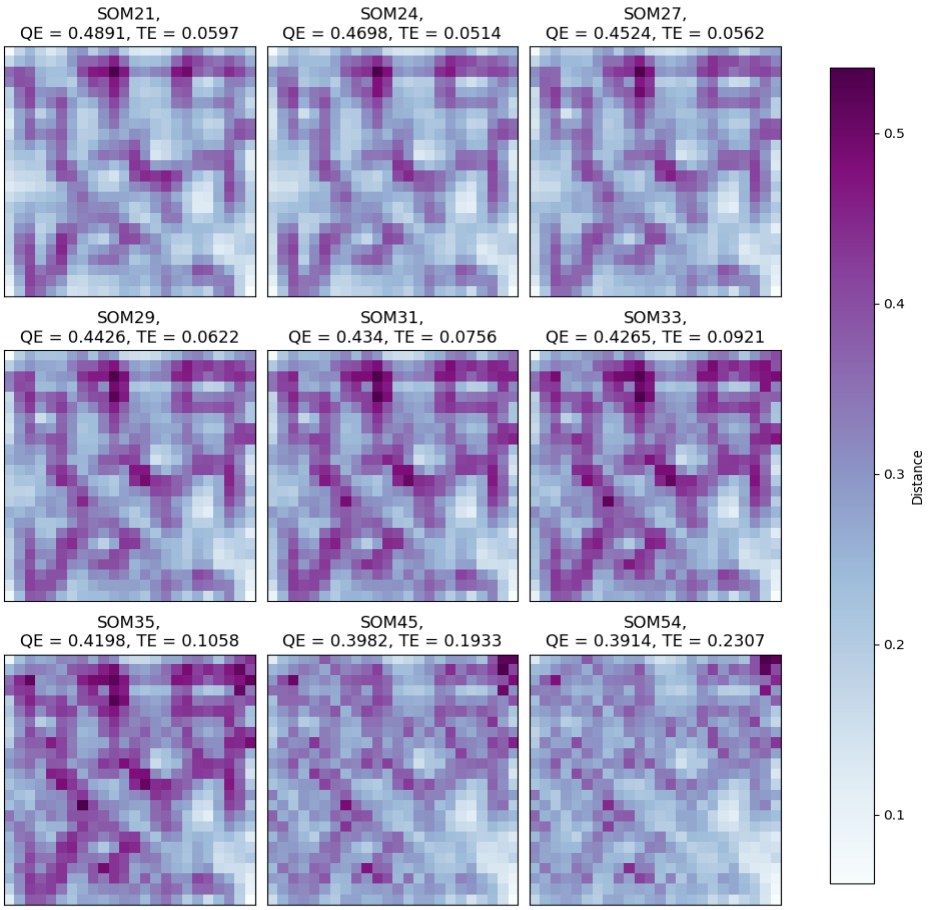
```
def train_SOM(SOM, train_data, learn_rate = .1, radius_sq = 1, lr_decay = .1, radius_decay = .1, epochs = 10):
    SOM_list = []
    TE_list = []
    QE_list = []
    learn_rate_0 = learn_rate
    radius_0 = radius_sq

    for epoch in np.arange(0, epochs):
        np.random.shuffle(train_data)
        for train_ex in train_data:
            g, h = find_BMU(SOM, train_ex)
            SOM = update_weights(SOM, train_ex, learn_rate, radius_sq, (g,h))
        learn_rate = learn_rate_0 * np.exp(-epoch * lr_decay)
        radius_sq = radius_0 * np.exp(-epoch * radius_decay)

        SOM_list.append(np.copy(SOM))  # copy保证添加进去的SOM不会跟着epoch继续迭代，否则会得到一个全部一样的SOM list
        TE_list.append(calculateTE(SOM,train_data))
        QE_list.append(calculateQE(SOM,train_data))

    return SOM, SOM_list, TE_list, QE_list
```

QE is a quantitative error, it means whether the SOM is a good representation of all training data. For each data point, calculate the distance between data nad its BMU, finally get average distance for all data. So the lower QE value is better.
TE is a topological error, it means whether similar data is well organized. For each data point, find its BMU1 and BMU2, if the Manhattan distance between two units bigger than 1, it is means the topology of this data is misrepresented, and finally calculate the error rate. So TE value also need to be low.
Define the plot_TE_QE function, to plot a curve based on the QE_list and TE_list obtained during training. After training, call this function to see how QE and TE change in epochs.



For example, in this Text SOM, QE is getting lower and lower, which means SOM is getting more and more understanding of the meaning of text, but TE increases significantly after the 30th epoch, which means the organizational relationship between units is not good, so SOM_list[30] should be a best choice.

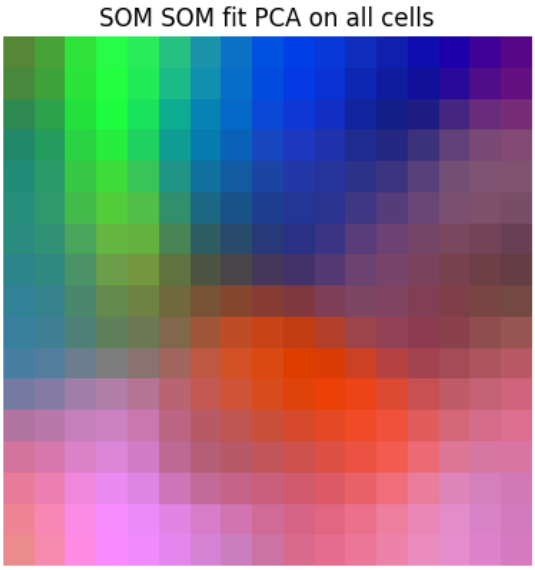
Also print U-matrix map to check the units organaztion.



Create a function to plot a SOM based on PCA , get the 3 principal components used as RGB values.

First way: Fit PCA on all cells of the SOM.

This method get 3 principal component directions based on the weights of all cells themselves, and then uses this PCA model to transform the vector of each cell, to see the values of each unit in these 3 main directions.



Second way: Fit PCA on the entire traindata and then use PCA on each cell.
This way will get 3 principal component based on the entrie train data, and use this PCA model to transfer the vector of each cell. So this way is to see the relationship between SOM cells and the entire dataset.

2.1.3 Fill text into SOM

In addition to save a list "textGrid_all" that fills all the data into their BMU, also saved another list "textGrid", only contains the most similar sentence within each cell.
So when using the search engine at the end, although all the sentences in this cell are kept by a variable, only the most similar sentence is printed, it will look like cleaner.

```
[ ] textGrid_all = []
    for i in range(SOM.shape[0]):
            row = []
            for j in range(SOM.shape[1]):
                row.append([])
            textGrid_all.append(row)

    for word, vector in zip(original_texts, train_data):
            bmu = find_BMU(SOM, vector)
            textGrid_all[bmu[0]][bmu[1]].append(word)
```

```
[ ] len(textGrid_all[0][1])

    26
```

```
[ ] textGrid_all[0][1]

    ['The greatByzantineandPersian Empires to the immediate north had a history of expansion and conflict. Despite their strength, however, neither desired to dominate Arabia. To those
    classical states, much of Arabia appeared as a backwater occupied by migratory and aggressive Arab tribes and offered no reason for them to turn their imperial ambitions southward.
    Few resources were produced in the region that suggested conquest would be worthwhile, even if western Arabia did play a role in the caravans of trade goods that traveled between
    east and west.',
     'In the very south of the Arabian Peninsula, in what is Yemen today, was a kingdom known asHimyar. Its rulers controlled some of the most fertile lands in the region. They built
    their state on agricultural produce, on luxury goods such as frankincense and myrrh, and on their role as intermediaries in both East African and Indian Ocean trade. The Himyarites
```

```
[ ] vector_index_mapping = {}
    for i in range(SOM.shape[0]):
        for j in range(SOM.shape[1]):
            vector_index_mapping[(i, j)] = []

    for index, (word, vector) in enumerate(zip(original_texts, train_data)):
        bmu = find_BMU(SOM, vector)
        vector_index_mapping[bmu].append(index)    # 将文本的索引（而不是文本本身）添加到对应的神经元
```

```
[ ] textGrid = []
    for i in range(SOM.shape[0]):
        for j in range(SOM.shape[1]):
            if len(vector_index_mapping[(i, j)]) > 0:
                closest_text_index = get_closest_text(i, j, SOM, vector_index_mapping, train_data)
                textGrid.append(original_texts[vector_index_mapping[(i, j)][closest_text_index]])
            else:
                textGrid.append('None')
```

2.1.4 Create a search function

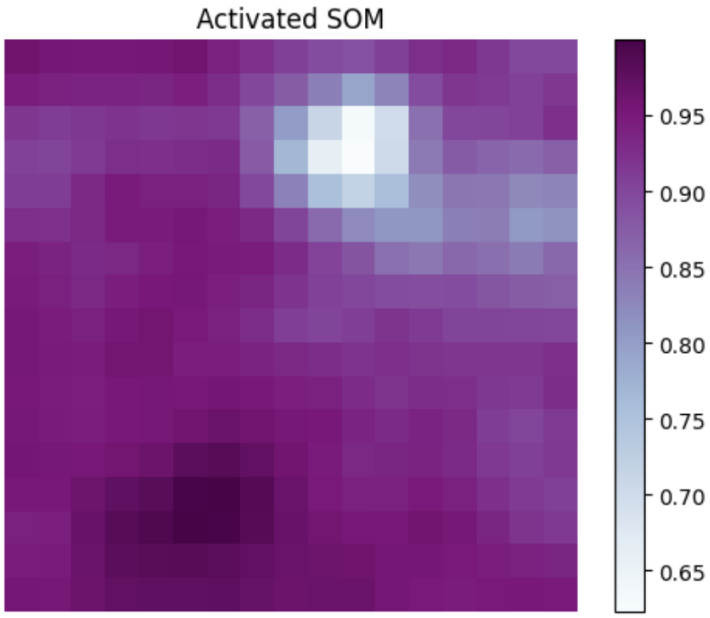
Create a function to receive the query sentence, preprocessed and vectorize it, find its BMU. Print the representative sentence of this unit through the "textGrid" lists, and return a list of all the sentences in this unit.
According to the dictionary constructed before, with the sentence as the key, get the image path, and show this picture. Returns the image_path, so that it can be used as query input to the Image SOM.

```
def query(SOM, query_word, textGrid, train_data):
    processedQuery = [preprocess_stemm(query_word)]
    query_vector = tfidf_vectorizer.transform(processedQuery)
    reduced_query_vec = svd_vectorizer.transform(query_vector)

    activatedSOM = activate(train_data, SOM, reduced_query_vec)
    bmu = find_BMU_flattened(SOM, reduced_query_vec)
    bmu_index = bmu[0] * SOM.shape[1] + bmu[1]
    bmu_sentence = textGrid[bmu_index]
    all_sentences_inbmu = textGrid_all[bmu[0]][bmu[1]]
    book_id, page_id, images = sentence_find_image.get(bmu_sentence, (None, None, None))

    fig = plt.figure()
    plt.imshow(activatedSOM, cmap=cm.BuPu, interpolation='nearest')
    plt.title('Activated SOM')
    plt.colorbar()
    plt.axis('off')
    plt.show()
    print('BMU:', bmu)
    print()
    print(f'There are {len(all_sentences_inbmu)} paragraphs in this unit, print the most similar one as Result Text:')
    print()
    print(textwrap.fill(bmu_sentence, width=170))

    if bmu_sentence == 'None':
        print('This unit is empty, please try another query.')
        return

    print(images)
    for i in images:
        query_img = img_reshape(i)
        plt.imshow(query_img)
        plt.title('Corresponding Image')
        plt.axis('off')
        plt.show()
    return bmu_sentence, images, all_sentences_inbmu
```

2.1.5 Example outcome

When using this function, will get result like that:

```
[25] query_word = 'Immediately behind the palace, he built a large classical pavilion and deco
     bmu_sentence, images, all_sentences_inbmu = query(SOM, query_word, textGrid, train_data)
```



BMU: (3, 10)

There are 47 paragraphs in this unit, print the most similar one as Result Text:

Theimperial cultwas a group of rites and practices that praised a deceased emperor's divine status. Emperors were often deified after they died, by order of their successors and with approval by the Senate. This formal process ofdeificationwas known asapotheosisand was extended to emperors who were remembered favorably . The process of deification had become so routine among later emperors that when the emperor Vespasian was dying, he is reported to have said, "Alas! I think I am becoming a god!"
['/content/drive/MyDrive/main_design/textbook/history/image_world_history_1/page_300_image_0.png']

2.2 Train Image SOM

Code: *https://github.com/UD-Skills-2023-24/github-codebook-RC11-23114486/blob/main/Final_Assignment_1_Julian/Image_SOM.ipynb*

2.2.1 Preprocess and Vectorize

Load images and create a new dictionary "image_find_sentence" to ensure that the corresponding sentence could be found by images.

Try two different model to vectorize images to feature vector: Xception and Mobilenet. And create a function to test them by inputing a same picture to find top 5 similar images in dataset.

```
def check_similarity(similarities, query_index, features, featureImagePairs):
    query_vector = features[query_index]
    nearest_neighbor_index = similarities[query_index].argsort()[-2]
    nearest_neighbor_similarity = similarities[query_index, nearest_neighbor_index]
    top5_indices = similarities[query_index].argsort()[-6:-1][::-1]
    top5_scores = similarities[query_index, top5_indices]

    query_img_path = featureImagePairs[query_index]['image']
    query_img = img_reshape(query_img_path)
    plt.figure(figsize=(2.8, 2.8))
    plt.imshow(query_img)
    plt.title('query image')
    plt.axis('off')
    plt.show()

    plt.figure(figsize=(15,3))
    for idx, (index, score) in enumerate(zip(top5_indices, top5_scores),1):
        path = featureImagePairs[index]['image']
        img = img_reshape(path)
        plt.subplot(1, 5, idx)
        plt.imshow(img)
        plt.title(f"Index:{index}\nSimilarity:{score:.2f}")
        plt.axis('off')

    plt.tight_layout()
    plt.show()
```

```
similarities_Xception = cosine_similarity(Xception_features)
similarities_Mobilenet = cosine_similarity(Mobilenet_features)
```

test_1 Xception

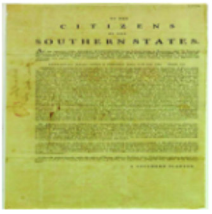
query image

test_1 Mobilenet

Index:113
Similarity:0.69

Index:617
Similarity:0.59

Index:549
Similarity:0.59



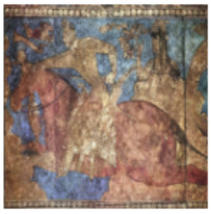
Index:439
Similarity:0.58

Index:877
Similarity:0.57

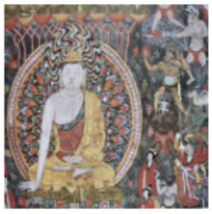
query image

test_2 Xception

Index:325
Similarity:0.67



Index:433
Similarity:0.65



Index:398
Similarity:0.64

Index:341
Similarity:0.61

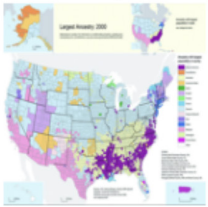
Index:369
Similarity:0.60

query image

test_2 Mobilenet

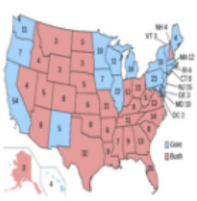
Index:632
Similarity:0.65

Index:1023
Similarity:0.65



Index:97
Similarity:0.64

Index:1012
Similarity:0.63

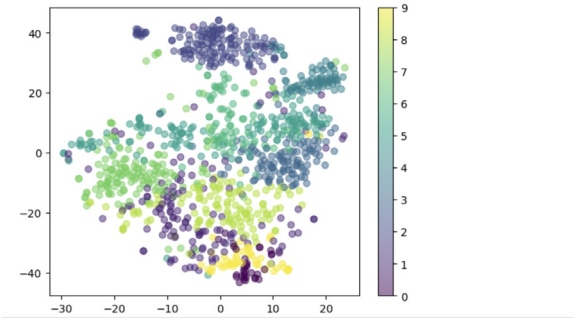


Index:575
Similarity:0.63

Using the Elbow Method to decide the optimal k, set 10 clustering centers for this dataset, assign 10 labels to the data by Kmeans, and then visualize the data using t-SNE to see if all the images are effectively separated.

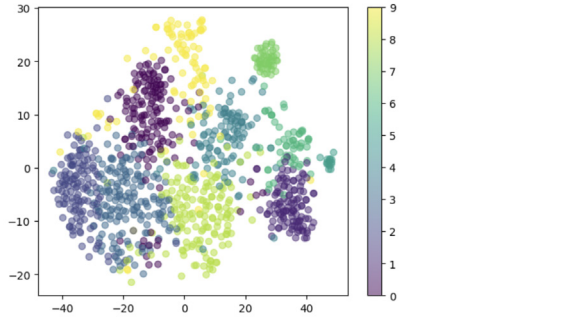
test_3 Xception

```
kmeans = KMeans(n_clusters = 10, n_init='auto').fit(Xception_features)

labels = kmeans.labels_

tsne = TSNE(n_components=2, random_state=42, perplexity=30)
features_reduced = tsne.fit_transform(Xception_features_np)

plt.scatter(features_reduced[:, 0], features_reduced[:, 1], c=labels, cmap='viridis', alpha=0.5)
plt.colorbar()
plt.show()
```



test_3 Mobilenet

```
kmeans = KMeans(n_clusters = 10, n_init='auto').fit(Mobilenet_features)

labels = kmeans.labels_

tsne = TSNE(n_components=2, random_state=42, perplexity=30)
features_reduced = tsne.fit_transform(Mobilenet_features_np)

plt.scatter(features_reduced[:, 0], features_reduced[:, 1], c=labels, cmap='viridis', alpha=0.5)
plt.colorbar()
plt.show()
```

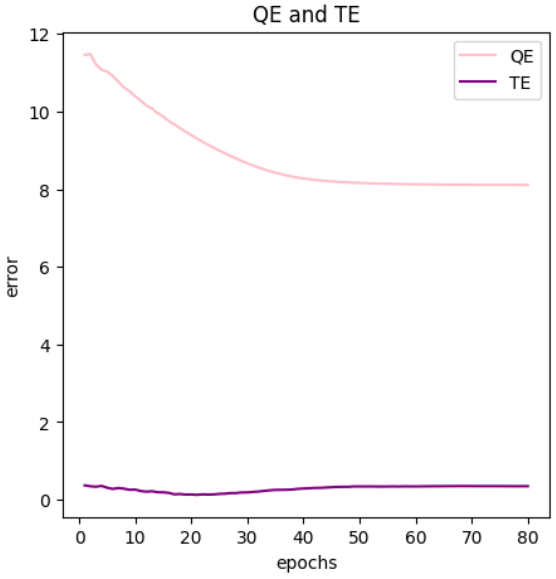


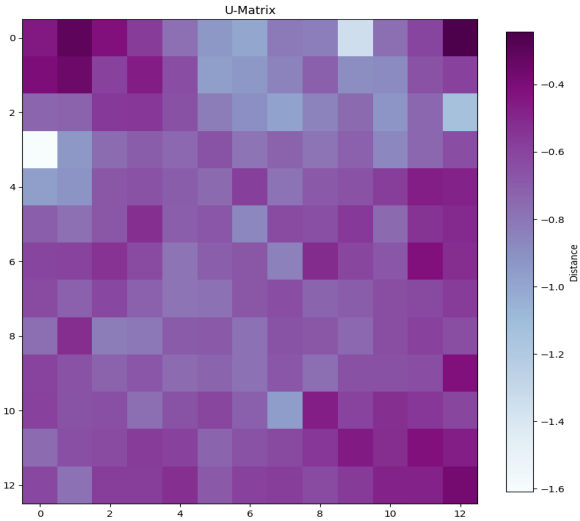
By comparison of similarity, the xception model seems to match images more accurately. But from test_3, Mobilenet's features are obviously classified better. Considering data are used for training SOM, perhaps better clustering will helpul, so it is decided to use this model to vectorize.

2.2.2 Train SOM

This part is same as the Text SOM, so I will not repeat the explanation, just show the results directly.

SOM = SOM_list[59]





2.2.3 Fill image into SOM

As same as the Text SOM, saved two list, "imageGrid_all" and "imageGrid", when using the search engine at the end, only show the most similar image.

2.2.4 Create a search function

Create a function to receive the query image, preprocessed and vectorize it, find its BMU. Show the representative image of this unit and return a list of all the image path in this unit. According to the dictionary constructed before, get the corresponding sentence by image path.

```
def query_image(img_path):
        if img_path.endswith('.mp4'):
            img_path = img_path.replace('mp4', 'jpg').replace('video', 'image')

        query_img = img_reshape(img_path)
        feature = processImage(img_path, model)
        activatedSOM = activate(train_data, SOM, feature)
        min_f = np.min(feature)
        max_f = np.max(feature)
        n_train_data = (feature-min_f)/(max_f-min_f)
        bmu = find_BMU(SOM, n_train_data)
        bmu_index = bmu[0] * SOM.shape[1] + bmu[1]
        bmu_image_path = imageGrid[bmu_index]
        bmu_image = img_reshape(bmu_image_path)
        all_images_inbmu = imageGrid_all[bmu[0]][bmu[1]]

        fig = plt.figure()
        plt.imshow(query_img)
        plt.title('Query Image')
        plt.axis('off')
        plt.show()
        plt.imshow(activatedSOM, cmap=cm.BuPu, interpolation='nearest')
        plt.title('Activated SOM')
        plt.colorbar()
        plt.axis('off')
        plt.show()
        print('BMU: ', bmu)
        print(f'There are {len(all_images_inbmu)} images in this unit, print the most similar one as Result Image')
        plt.imshow(bmu_image)
        plt.title('Result Image')
        plt.axis('off')
        plt.show()
        display(HTML('<b>corresponding text in book next to image:<b>'))

        texts = image_find_text.get(bmu_image_path, 'No text found for this image')
        for text in texts:
            print(textwrap.fill(text, width=170))
            print()
        return bmu_image_path, all_images_inbmu
```

2.2.5 Example outcome

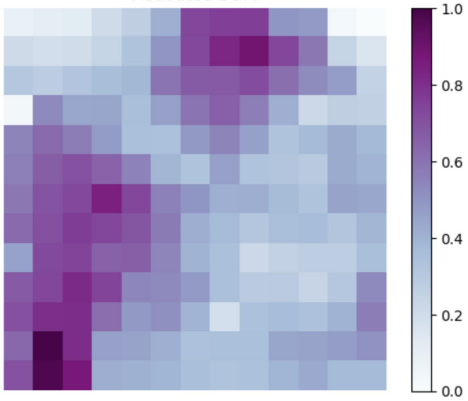
When using this function, will get result like that:

```
[ ] img_path = '/content/drive/MyDrive/Materials/image/painting_for_som/49.jpg'
    bmu_image_path, all_images_inbmu = query_image(img_path)
```

```
1/1 [==============================] - 0s 188ms/step
```

Query Image

Activated SOM




```
BMU:  (12, 1)
```

```
There are 10 images in this unit, print the most similar one as Result Image
```

Result Image

corresponding text in book next to image:

The 1920s was a time of dramatic change in the United States. Many young people, especially those living in big cities, embraced a new morality that was much more permissive than that of previous generations. They listened to jazz music, especially in the nightclubs of Harlem. Although prohibition outlawed alcohol, criminal bootlegging and importing businesses thrived. The decade was not a pleasure cruise for everyone, however; in the wake of the Great War, many were left awaiting the promise of a new generation.

Many Americans were disillusioned in the post-World War I era, and their reactions took many forms. Rebellious American youth, in particular, adjusted to the changes by embracing anew moralitythat was far more permissive than the social mores of their parents. Many young women of the era shed their mother's morality and adopted the dress and mannerisms of aflapper, the Jazz Age female stereotype, seeking the endless party. Flappers wore shorter skirts, shorter hair, and more makeup, and they drank and smoked with the boys . Flappers' dresses emphasized straight lines from the shoulders to the knees, minimizing breasts and curves while highlighting legs and ankles. The male equivalent of a flapper was a "sheik," although that term has not remained as strong in the American vernacular. At the time, however, many of these fads became a type of conformity, especially among college-aged youths, with the signature bob haircut of the flapper becoming almost universal—in both the United States and overseas.

2.3 Train Video SOM1

Code: *https://github.com/UD-Skills-2023-24/github-codebook-RC11-23114486/blob/main/Final_Assignment_1_Julian/Video_SOM1_text.ipynb*

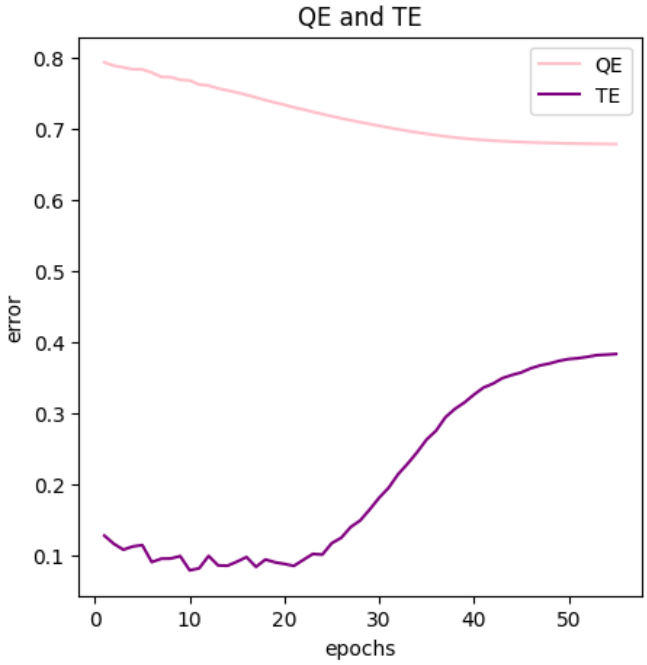
2.3.1 Preprocess and Vectorize

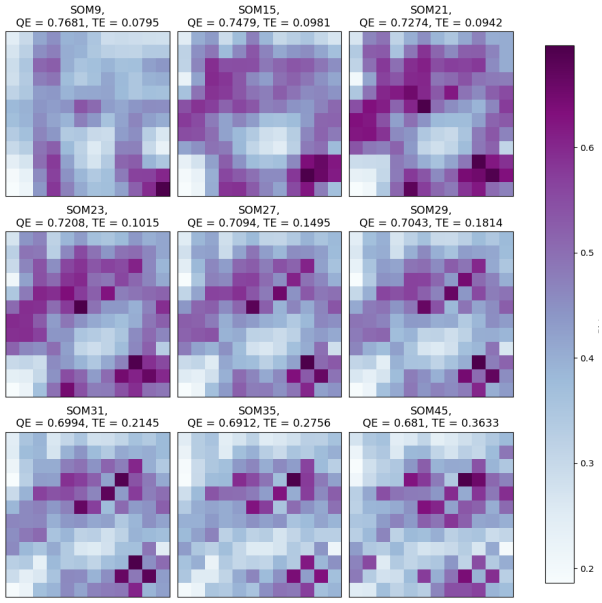
Use the fuyu model to get a text description of each video frame. Load text from csv file, create a dictionary, set sentence as key and the video frame path as value, to ensure the video and the frame could be found by sentence.Preprocess and vectorize them by the same method I used in Text SOM.

2.3.2 Train SOM

This part is same as the Text SOM, just show the outcome.

SOM = SOM_list[23]





2.3.3 Fill text into SOM

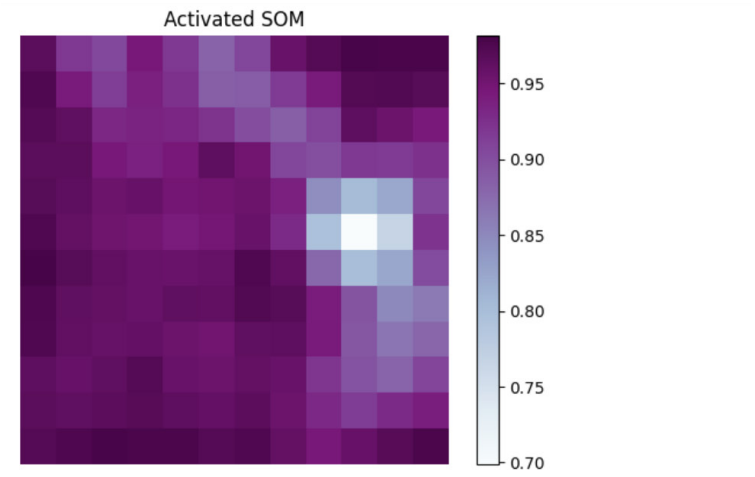
As same as the Text SOM, saved two list, "textGrid_all" and "texGrid", when using the search engine at the end, only show the most similar video.

2.2.4 Create a search function

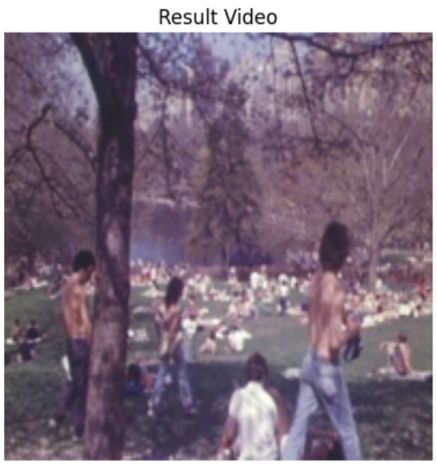
Create a function to receive the query sentence, preprocessed and vectorize it, find its BMU. Get the video frame path through the "sentence_find_frame" dictionary, show the representative sentence and frame of this unit and return a list of all the video path in this unit.

2.3.5 Example outcome

When using this function, will get result like that:




```
BMU:  (5, 9)

There are 36 video in this unit, print the most similar one as Result Video:
```




```
/content/drive/MyDrive/Materials/video/skill_final_poc/Picnic_area_Person/frame-133952.mp4

The image features a park where many people are gathered, some standing and some sitting. An
```

Create a function to play the video:

```
Text2Video_model.playVideo(video_path)
```

```
Moviepy - Building video /content/temp_video.mp4.
MoviePy - Writing audio in temp_videoTEMP_MPY_wvf_snd.mp3
MoviePy - Done.
Moviepy - Writing video /content/temp_video.mp4

t: 100%|██████████| 300/301 [00:08<00:00, 40.60it/s, now=None]WARNING:py.wa
  warnings.warn("Warning: in file %s, "%(self.filename)+

Moviepy - Done !
Moviepy - video ready /content/temp_video.mp4
```

2.4 Train Video SOM2

Code: *https://github.com/UD-Skills-2023-24/github-codebook-RC11-23114486/blob/main/Final_Assignment_1_Julian/Video_SOM2_image.ipynb*

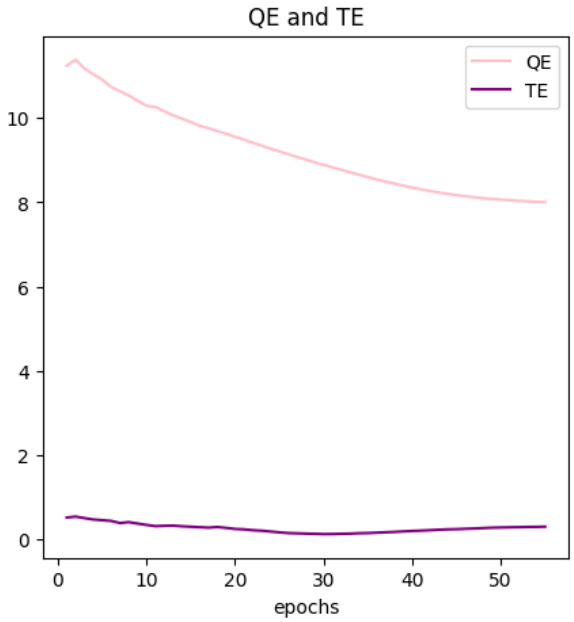
2.4.1 Preprocess and Vectorize

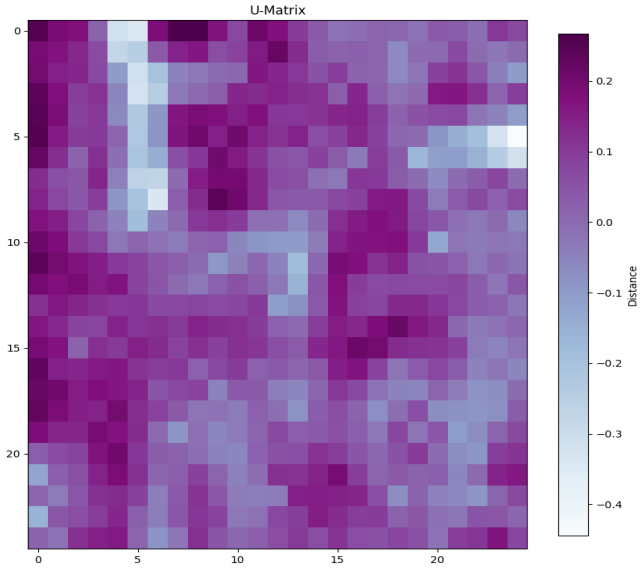
Use one frame to represent the video. Load frame, preprocess and vectorize them by the same method I used in Image SOM.

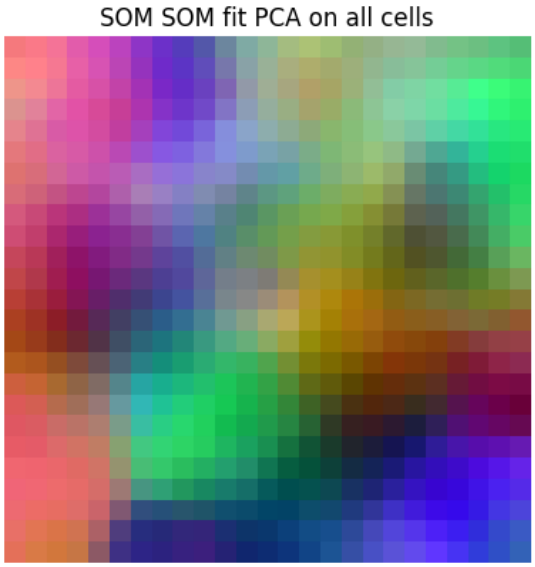
2.4.2 Train SOM

This part is same as the Text SOM, just show the outcome.

SOM = SOM_list[35]

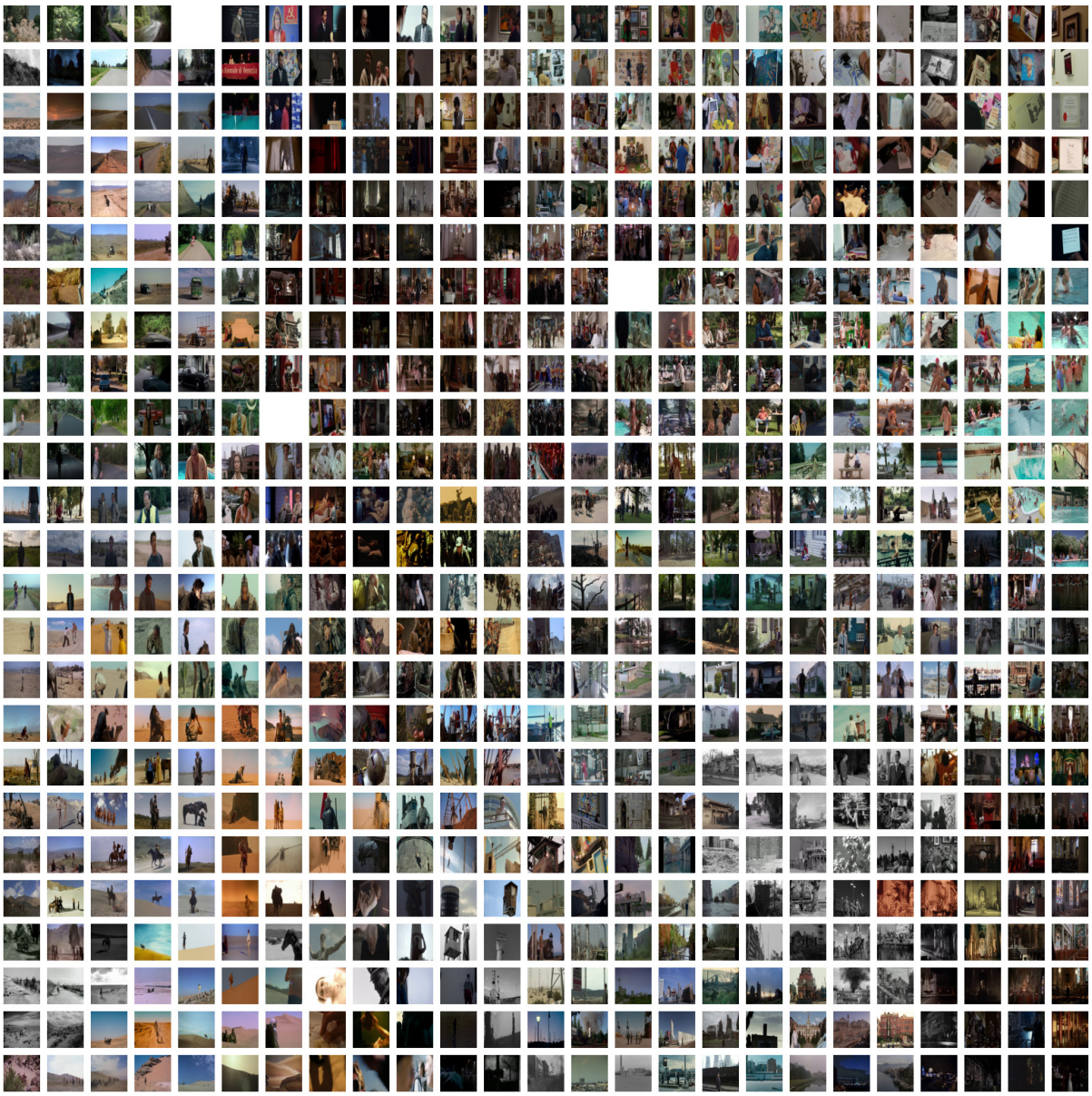






2.4.3 Fill image into SOM

As same as the Text SOM, saved two list, "imageGrid_all" and "imageGrid", when using the search engine at the end, only show the most similar video.



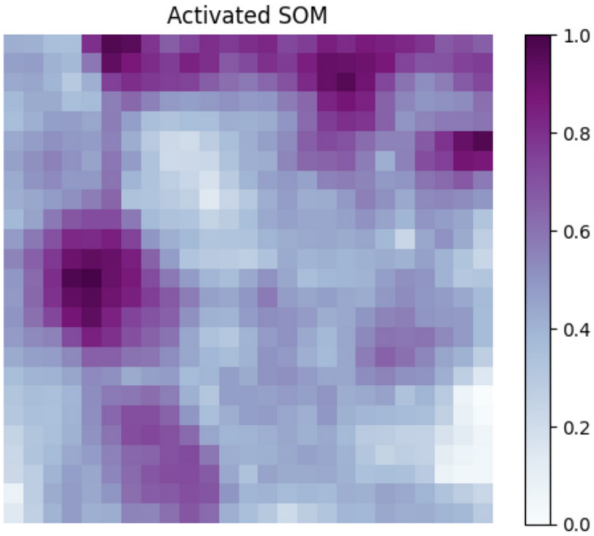
2.2.4 Create a search function

Create a function to receive the query image, preprocessed and vectorize it, find its BMU. Get the frame path and replace'.jpg' with '.mp4' to get the video frame path, only show the representative frame of this unit, and return a list of all the video path in this unit.

2.4.5 Example outcome

When using this function, will get result like that:


```
1/1 [==============================] - 0s 90ms/step
```




```
BMU:  (0, 5)

There are 17 video in this unit, print the most similar one as Result Video:
```


```
Video path: /content/drive/MyDrive/Materials/video/skill_final_poc/Conference_center_Person/frame-072560.mp4
```

```
Moviepy - Building video /content/temp_video.mp4.
MoviePy - Writing audio in temp_videoTEMP_MPY_wvf_snd.mp3
MoviePy - Done.
Moviepy - Writing video /content/temp_video.mp4

Moviepy - Done !
Moviepy - video ready /content/temp_video.mp4
```

If want to see all the videos in this BMU:

```
all_videos_path = Image2Video_model.showalldatainBMU(all_videos_inbmu)
```

3. Encapsulate into class

Code: *https://github.com/UD-Skills-2023-24/github-codebook-RC11-23114486/blob/main/Final_Assignment_1_Julian/Materials/SkillFinal/querysom_model.py*

Add a new search function to deal with 'video finding video', when input a video, extract a certain frame and save as jpg file, then it could be input into Video SOM2 to find video.

```
[ ] video_path = '/content/drive/MyDrive/Materials/video/search_baseon_object/markert_outdoor/frame-147464.mp4'
    output_path = '/content/drive/MyDrive/Materials/SkillFinal/SOM_VideoImage/query1.jpg'
    frame_number = 200
```

```
[ ] frame = Video2Video_model.extract_VideoFrame(video_path, output_path, frame_number)
```

```
    Already saved the frame
```

```
[ ] video_path, bmu_image_path, text, all_videos_inbmu = Video2Video_model.query(frame)
```

Code: *https://github.com/UD-Skills-2023-24/github-codebook-RC11-23114486/blob/main/Final_Assignment_1_Julian/Final_Search_Engine.ipynb*

Encapsulate the 5 search functions into 5 classes, they have a same fuction called 'query', so only need to execute one line, like: Text2Text_model.query():
And store them in a .py file to simplify the search engine. it means 5 different search path:

From Video to Video,
From Text to Video,
From Image to Video,
From Text to Text,
From Image to Image.

So, to use this 'search engine', just load model like this:

```
[4] from querysom_model import (FromImageToImage, FromImageToVideo, FromTextToText, FromTextToVideo, FromVideoToVideo)
```

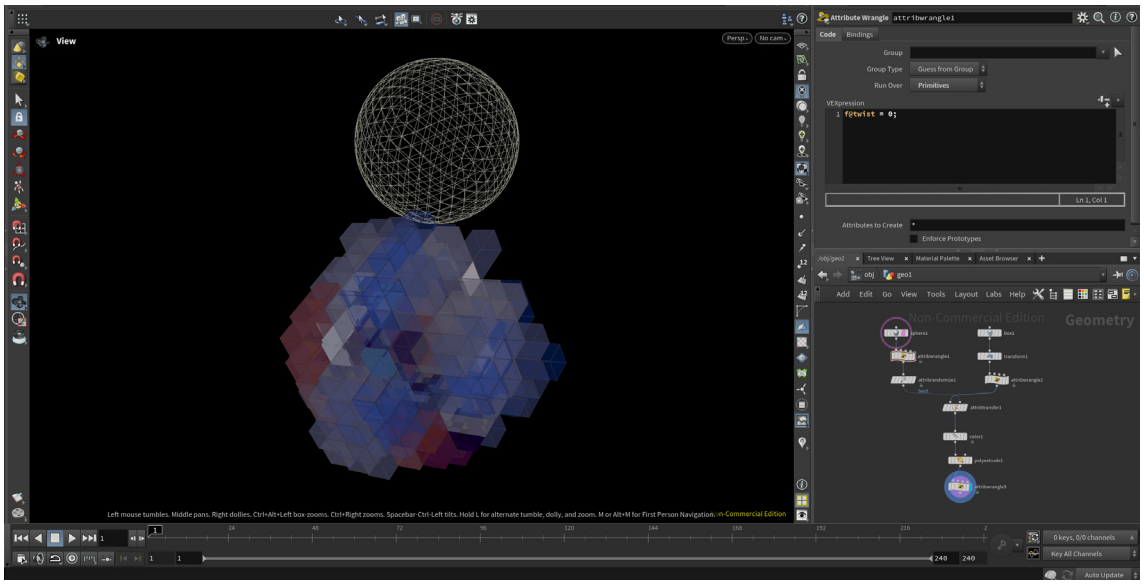
```
[5] path = '/content/drive/MyDrive/Materials/SkillFinal/SOM_Image'
    Image2Image_model = FromImageToImage(path)

    path = '/content/drive/MyDrive/Materials/SkillFinal/SOM_VideoImage'
    Image2Video_model = FromImageToVideo(path)

    path = '/content/drive/MyDrive/Materials/SkillFinal/SOM_Text'
    Text2Text_model = FromTextToText(path)

    path = '/content/drive/MyDrive/Materials/SkillFinal/SOM_VideoText'
    Text2Video_model = FromTextToVideo(path)

    path = '/content/drive/MyDrive/Materials/SkillFinal/SOM_VideoImage'
    Video2Video_model = FromVideoToVideo(path)
```

Then start searching.

FINAL_ASSIGNMENT_2

Tutor: Joris Putteneers

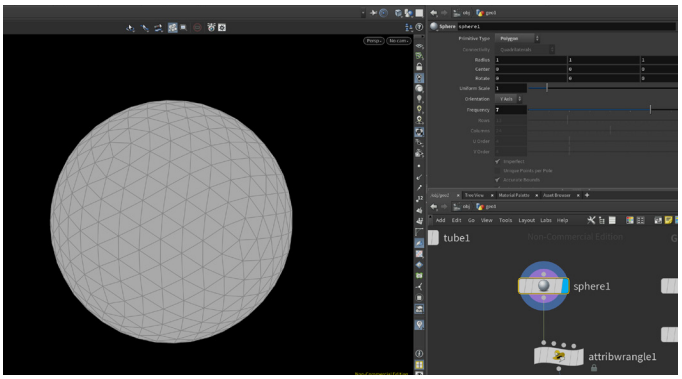
1 Houdini Fundamentals



Houdini Fundamentals: Setup 1 Overview

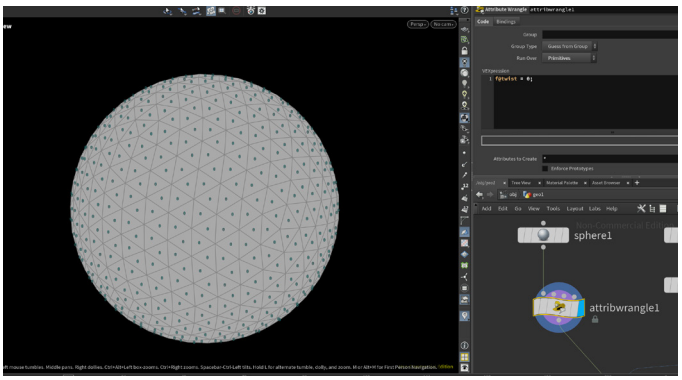
1. Sphere SOP

The "sphere" is of primitive type: polygon. One of the basic geometric data types that contains points, primitives of type polygon, and vertices.



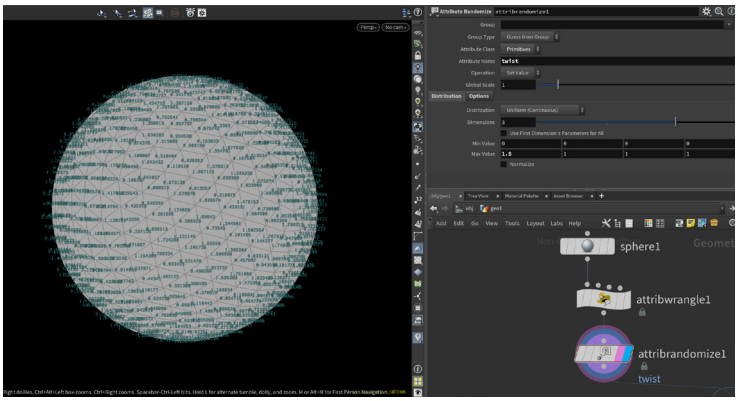
2. Attribute Wrangle SOP

Create an "Attribute Wrangle"SOP, where I add a "twist" attribute to the primitives, by the "f@", it means a float type. This one add value 0 to every primitive.



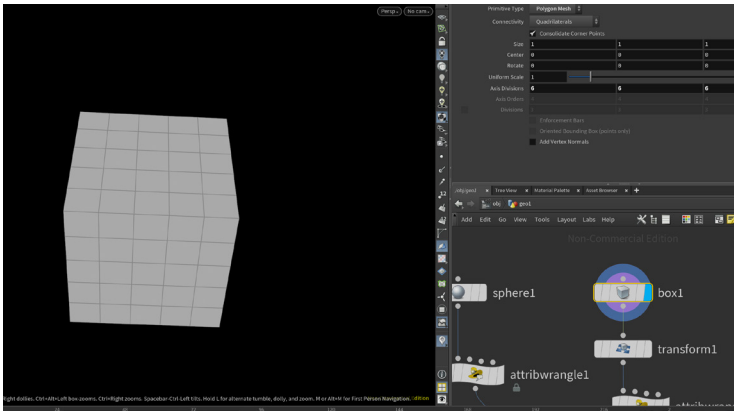
3. Attribute Randomize SOP

Randomized the value of weight into 0-1.8. The purpose of this step is to differentiate the results after attribute transfer.



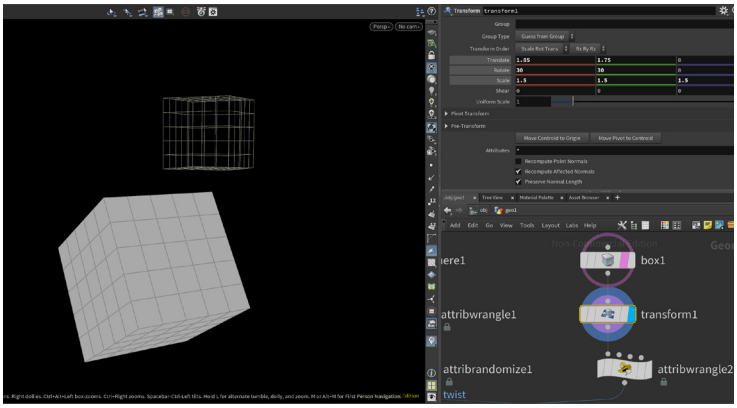
4. Box SOP

Create an Box SOP. One of the basic geometric data types that contains points, primitives of type polygon, and vertices.



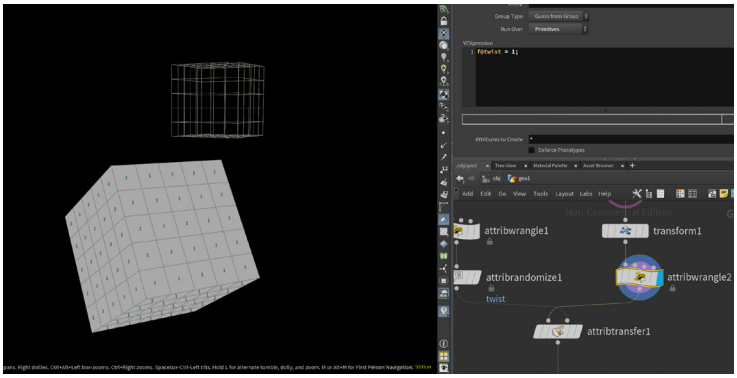
5. Transform SOP

It is a duplicate of the original box. This step is mainly expected to adjust the final extrusion Angle by rotate. The distance and rotate between this duplicate and the sphere will dictate the final extrusion value.



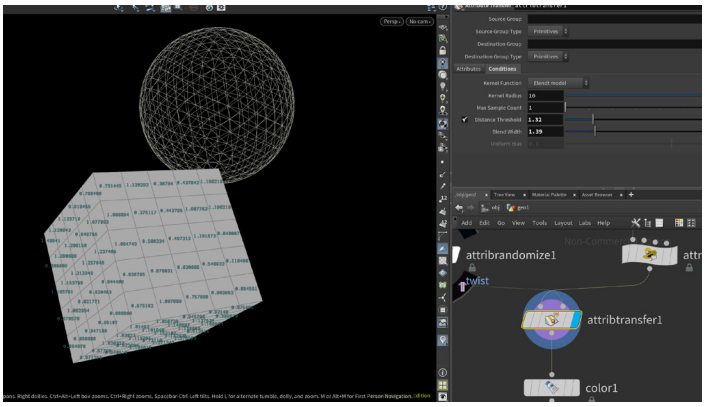
6. Attribute Wrangle SOP

Create an "Attribute Wrangle" SOP, where I add a "twist" attribute to the primitives. This one add value 1 to every primitive.



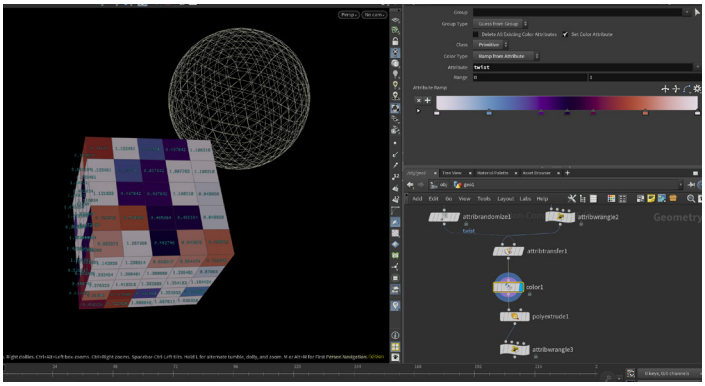
7. Attribute Transform SOP

Create an "Attribute Transfer" SOP, which used to transfer randomized weight to the transferred Box. By doing so, the weight on the enlarge box can change between 0 to 1.8, which can be done by sliding the bar "Distance Threshold" and "Blending With".



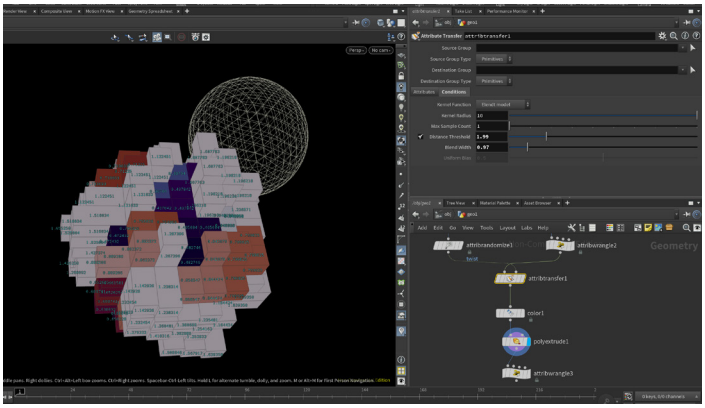
8. Color SOP

Create an "Color" SOP, where we add color to the box based on the value of "twist" on each primitive.



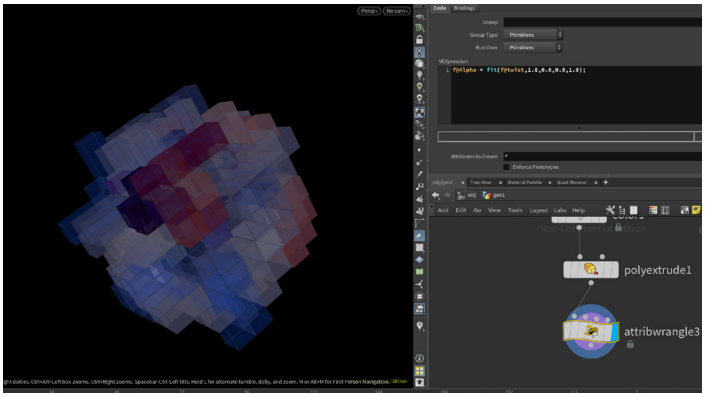
9. Polygon Extrude SOP

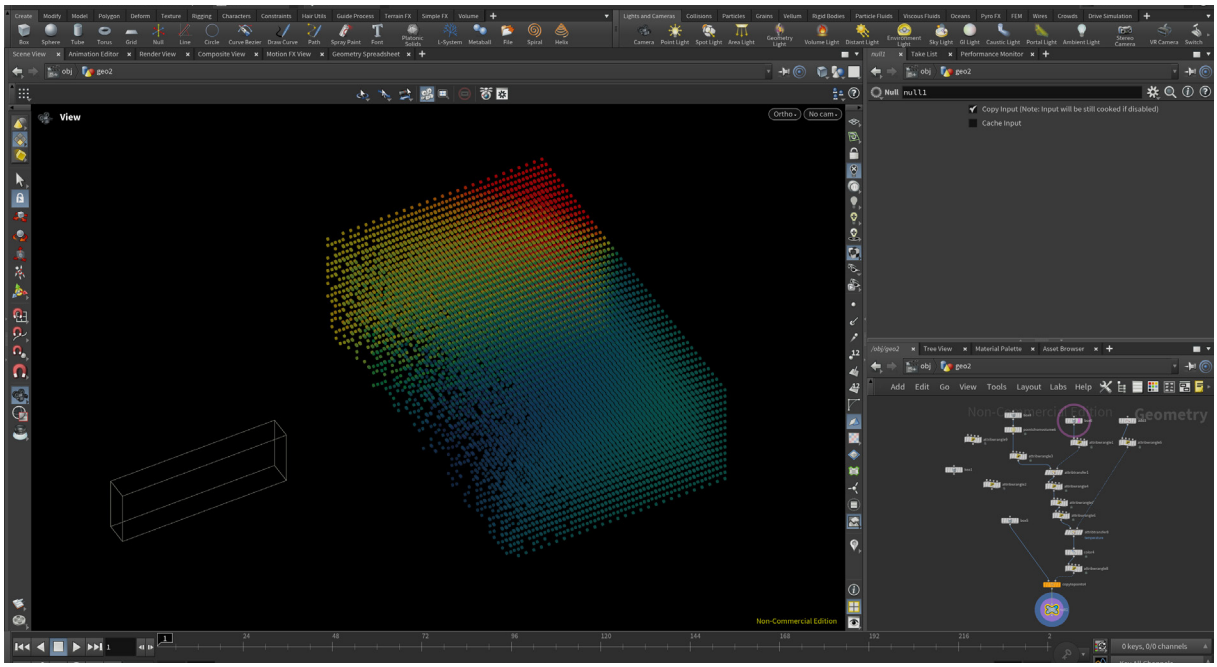
Create an "Polygon Extrude" SOP, which used to extrude inwards each primitive on the box generally based on the value of "twist".



10. Attribute Wrangle SOP

Create an "Attribute Wrangle" SOP, where we add a "Alpha" attribute to the primitives by "f@". Value of the Alpha of each primitive will be reversed according to the value of the twist property. The larger the value of twist, the more transparent it will be.

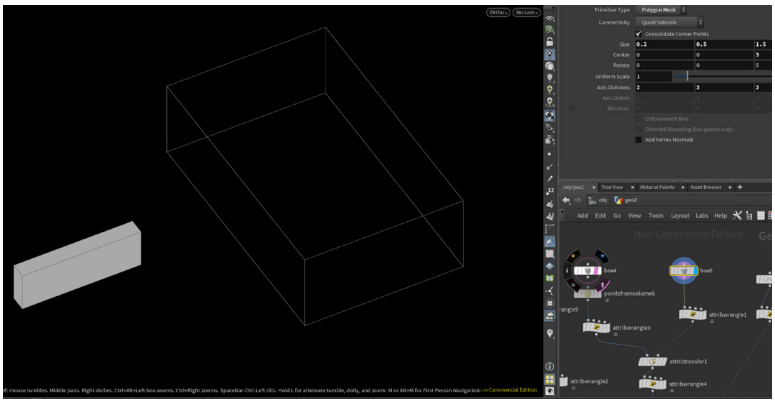




Houdini Fundamentals: Setup 2 Overview

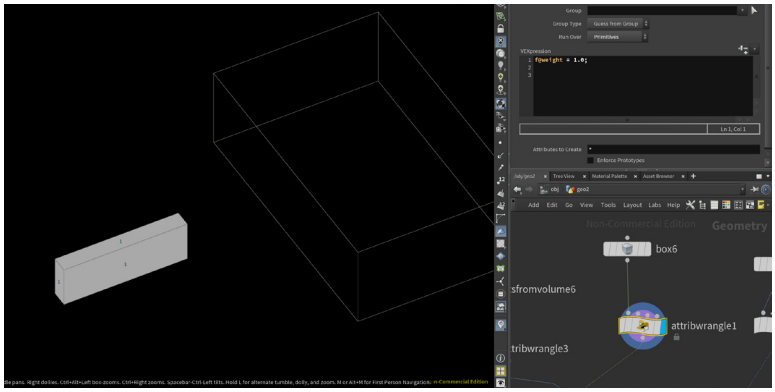
Based on the project written in class, the room temperature is simulated and measured and the heat map is displayed. It is assumed that there is wind blowing into the room at the "window", and the range of wind blowing into the room is simulated by the change of particle distribution density.

1. Box SOP



Create an Box SOP, and change the center position so that it is next to "window". One of the basic geometric data types that contains points, primitives of type polygon, and vertices.

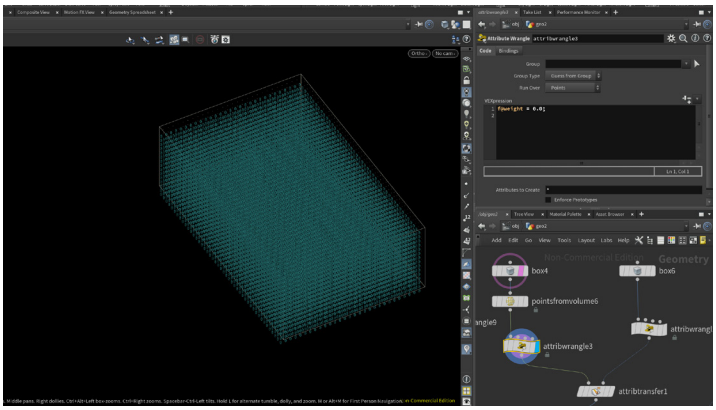
2. Attribute Wrangle SOP



Create an "Attribute Wrangle"SOP, where I add a "weight" attribute to the primitives, by the "f@", it means a float type. This one add value 1.0 to every primitive.

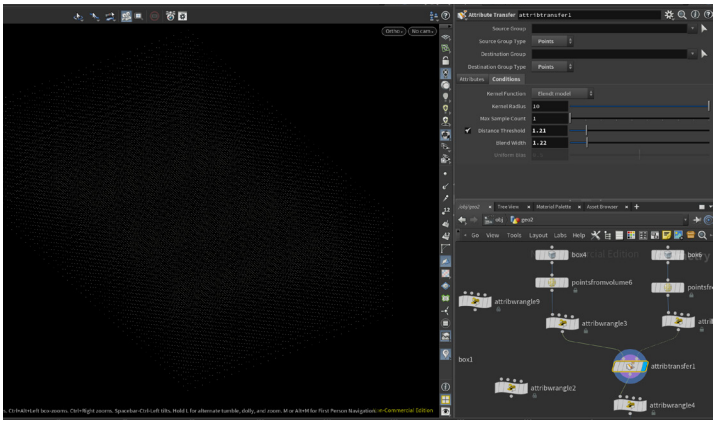
3. Attribute Wrangle SOP

Create an "Attribute Wrangle"SOP, where I add a "weight" attribute to all of the points. This one add value 0.0 to every points.



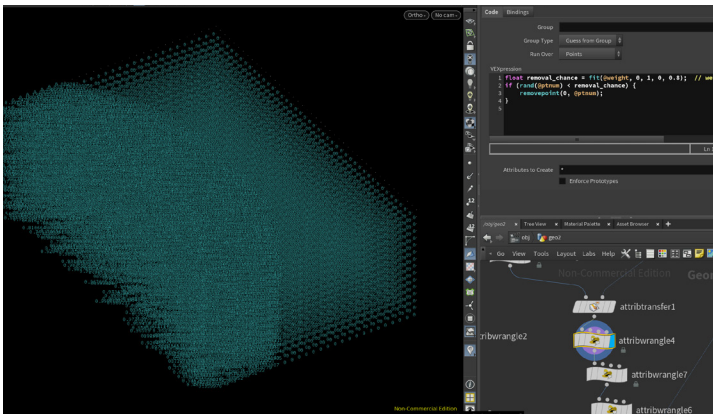
4. Attribute Transform SOP

Create an "Attribute Transfer" SOP, which used to transfer 1.0 weight to the points. By doing so, the weight on the points can change between 0 to 1.



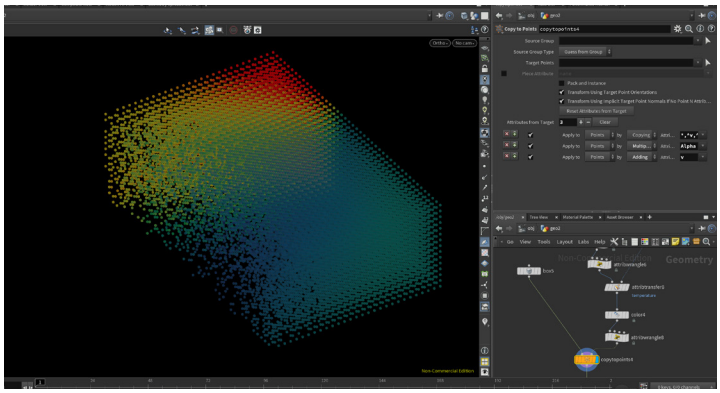
5. Attribute Wrangle SOP

The removal rate is determined according to the value of weight. The higher the value of weight, the higher the probability of the point being removed.



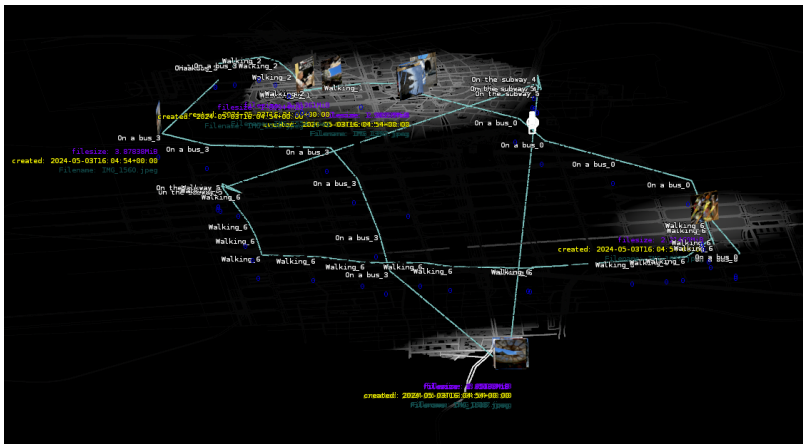
3. Copytopoints SOP

Copy box to each point, instantiate the points, and see the final effect.

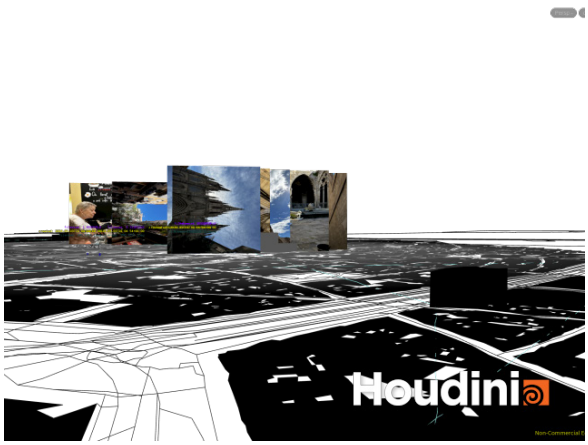


2 Visualizing GPS and Image Metadata

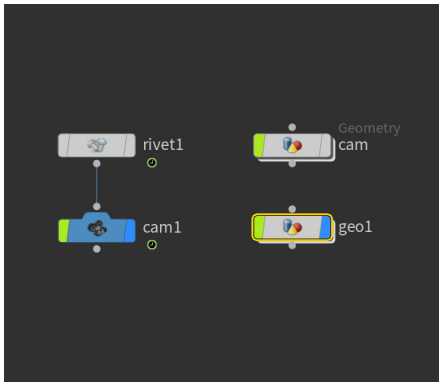
In this assignment, I used the travel track and mobile phone photos of my trip to Barcelona. The path, city map and images with geographic information are converted respectively. Import data and generate a series of images depicting travels.

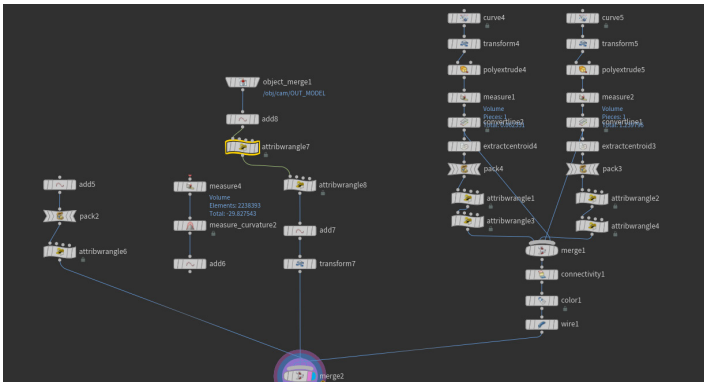


Add a transform node and transform it to face the camera.



3 video to 3D Model





The topic of our main design is knowledge and power mechanism, which mainly translates architecture through text materials. Part of the research focuses on existing buildings that are considered authoritative in history. So this assignment will download and generate the Acropolis of Athens and associate the "authoritative text" for the temple with the model.

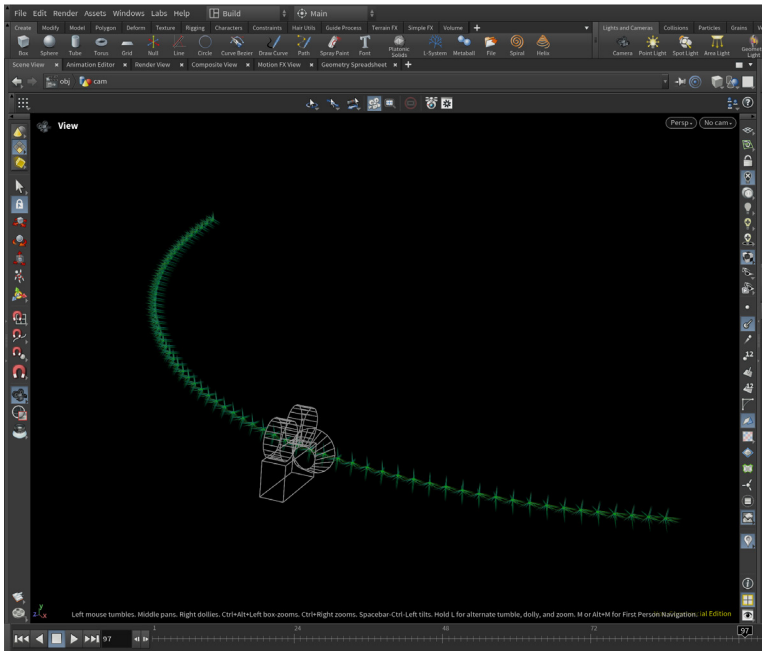
This assignment is divided into 4 parts. The first part is to generate a photogrammetry model from a video scraped from YouTube. The second step is to reconstruct the camera path. The third step is to mark some information related to the project, including the volume and text description of the main buildings. The fourth step is to establish the point cloud based on the model and make some adjustments.

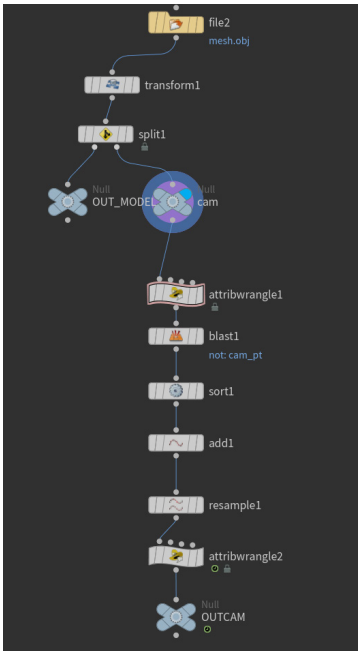
First Step

```
In [12]: url ="https://www.youtube.com/watch?v=wxN1T1uxQ2g"
         yt = YouTube(url)

In [13]: for stream in yt.streams:
             print(stream)

<Stream: itag="18" mime_type="video/mp4" res="360p" fps="24fps" vcodec="avc1.42001E" acodec="mp4a.40.2" progressive="True" type="video">
<Stream: itag="22" mime_type="video/mp4" res="720p" fps="24fps" vcodec="avc1.64001F" acodec="mp4a.40.2" progressive="True" type="video">
<Stream: itag="137" mime_type="video/mp4" res="1080p" fps="24fps" vcodec="avc1.640028" progressive="False" type="video">
<Stream: itag="248" mime_type="video/webm" res="1080p" fps="24fps" vcodec="vp9" progressive="False" type="video">
<Stream: itag="136" mime_type="video/mp4" res="720p" fps="24fps" vcodec="avc1.4d401f" progressive="False" type="video">
<Stream: itag="247" mime_type="video/webm" res="720p" fps="24fps" vcodec="vp9" progressive="False" type="video">
<Stream: itag="135" mime_type="video/mp4" res="480p" fps="24fps" vcodec="avc1.4d401e" progressive="False" type="video">
<Stream: itag="244" mime_type="video/webm" res="480p" fps="24fps" vcodec="vp9" progressive="False" type="video">
<Stream: itag="134" mime_type="video/mp4" res="360p" fps="24fps" vcodec="avc1.4d401e" progressive="False" type="video">
<Stream: itag="243" mime_type="video/webm" res="360p" fps="24fps" vcodec="vp9" progressive="False" type="video">
<Stream: itag="133" mime_type="video/mp4" res="240p" fps="24fps" vcodec="avc1.4d4015" progressive="False" type="video">
<Stream: itag="242" mime_type="video/webm" res="240p" fps="24fps" vcodec="vp9" progressive="False" type="video">
<Stream: itag="160" mime_type="video/mp4" res="144p" fps="24fps" vcodec="avc1.4d400c" progressive="False" type="video">
<Stream: itag="278" mime_type="video/webm" res="144p" fps="24fps" vcodec="vp9" progressive="False" type="video">
<Stream: itag="139" mime_type="audio/mp4" abr="48kbps" acodec="mp4a.40.5" progressive="False" type="audio">
<Stream: itag="140" mime_type="audio/mp4" abr="128kbps" acodec="mp4a.40.2" progressive="False" type="audio">
<Stream: itag="249" mime_type="audio/webm" abr="50kbps" acodec="opus" progressive="False" type="audio">
<Stream: itag="250" mime_type="audio/webm" abr="70kbps" acodec="opus" progressive="False" type="audio">
<Stream: itag="251" mime_type="audio/webm" abr="160kbps" acodec="opus" progressive="False" type="audio">

In [14]: video = yt.streams.get_by_itag(137)
         video.download()

Out[14]: 'E:\\UD 2023-2024\\RC11_Skill\\Joris_Houdini\\04_visualize_data\\Everything Everywhere All At Once  Official Trailer HD  A24.mp4'

In [15]: frame_no = 0

         cap = cv2.VideoCapture("Everything Everywhere All At Once  Official Trailer HD  A24.mp4")

In [16]: while (cap.isOpened()):
             ret, frame = cap.read()
             if(frame_no % 10 == 0):
                 name =  f"frame_{frame_no}.jpg"
                 target = os.path.join(imagepath,name)
                 cv2.imwrite(target, frame)
             frame_no +=1
```

Examine downloadable video streams, download the video and save pictures every 10 frames. Import consecutive frames into RealityCapture and generate as a model.

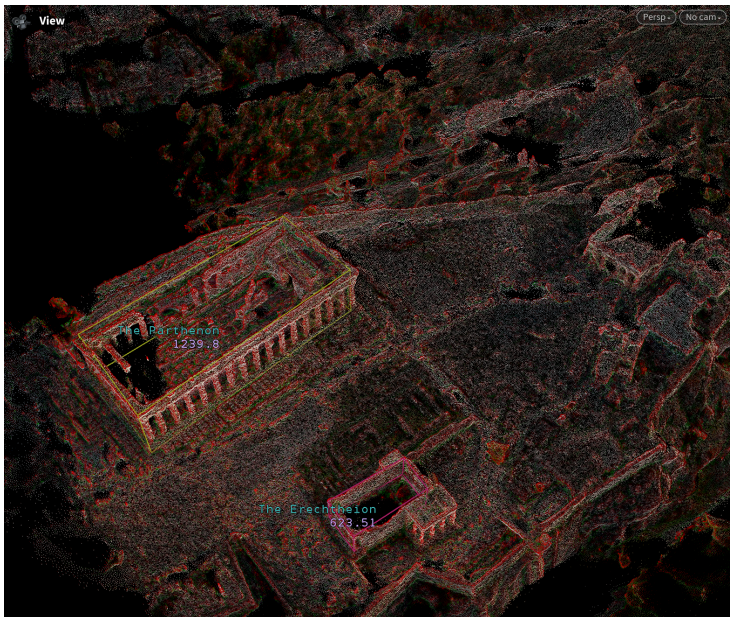
Second Step

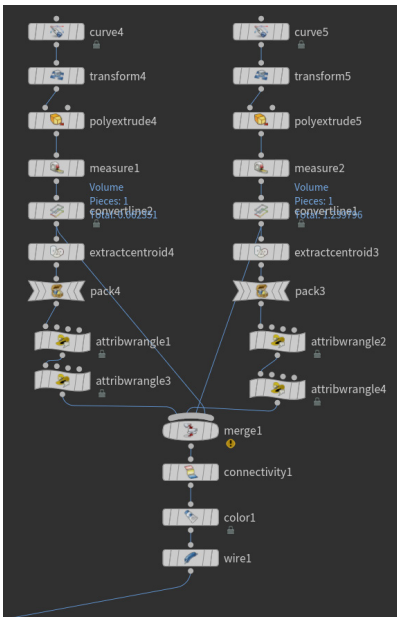




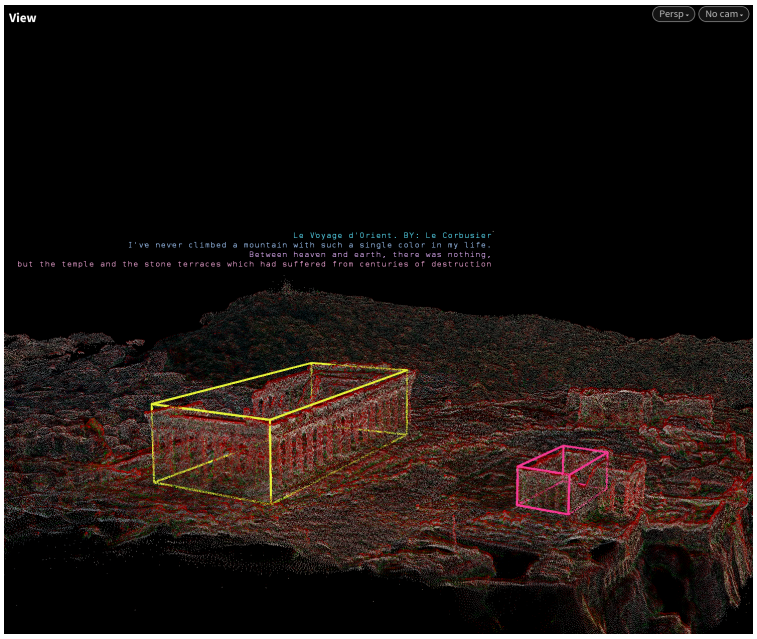
This step mainly deals with reconstruct the camera path in raw files. First, the imported data is separated so that the camera path could be processed separately. Use attribute wrangle1 to simplify the raw data and reorient the normal by finding centroid. Using "blast1" and "sort1" to remove useless points and reorder them, then resample, end up with a continuous and smooth camera path.

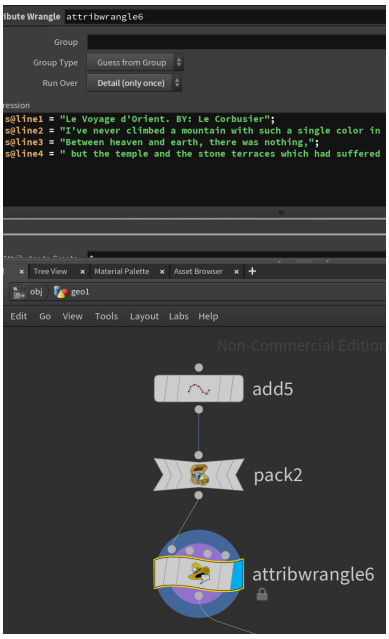
Third Step





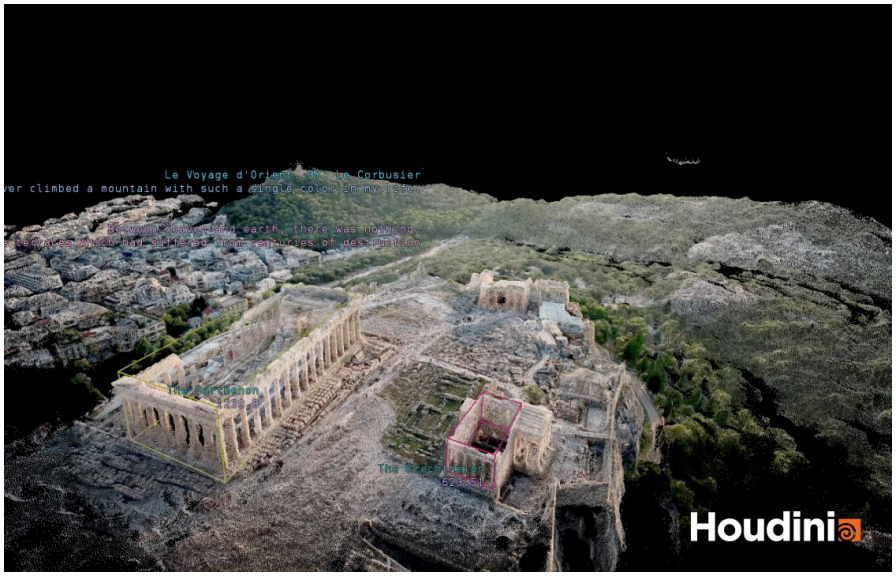
This part determines the approximate volume of the temple by determining the base area of the temple from the top view, followed by polyextrude, and then measuring the volume of the extruded volume. The volume is displayed in different colors based on the class attribute assigned by connectivity and is converted into wire.Visualizing the volume by adding attributes as marker.

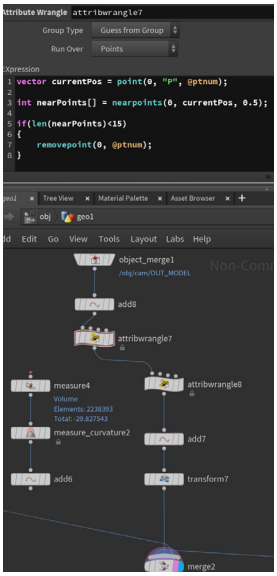




And add a point, package it and assign attributes to it, visualize the text in the view through marker. So it will display a description that is usually considered authoritative.

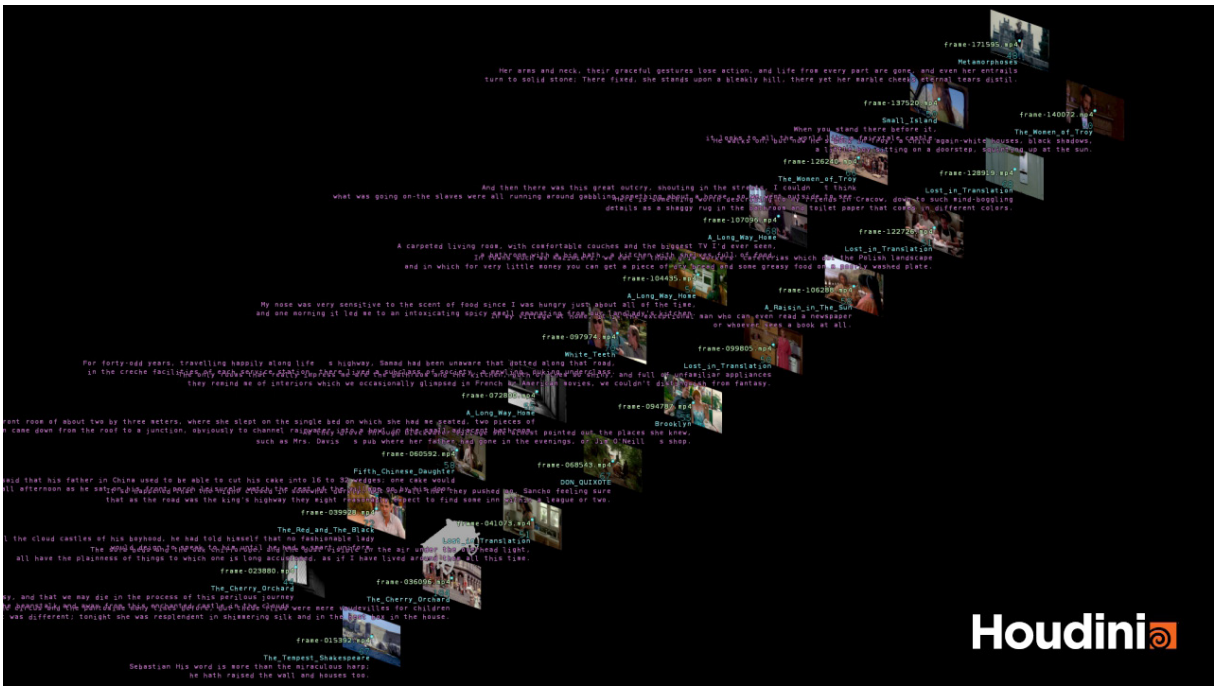
Fourth part





In this step, after transforming the base model into a point cloud, the denseness of the point cloud is calculated and the sparse part of the point cloud is removed to obtain the final point cloud model.

4 Visualizing JSON in Houdini



In term 1, I once tried a project about immigration, studied literature works about migration and matched them with videos, observed people's real activities, and tried to re-learn what immigration is instead of an empty definition.
So for this visualisation assignment, I tried to visualize the material at that time. I collected some video materials through the high-frequency scenes and high-frequency objects in immigrant literature. obtaining their frames, transforming the buildings in individual frames into models, and then printing out the original sentences in the literature and observing their association with the video.And try to convert the frame to 3D, extracting some of the objects in films.

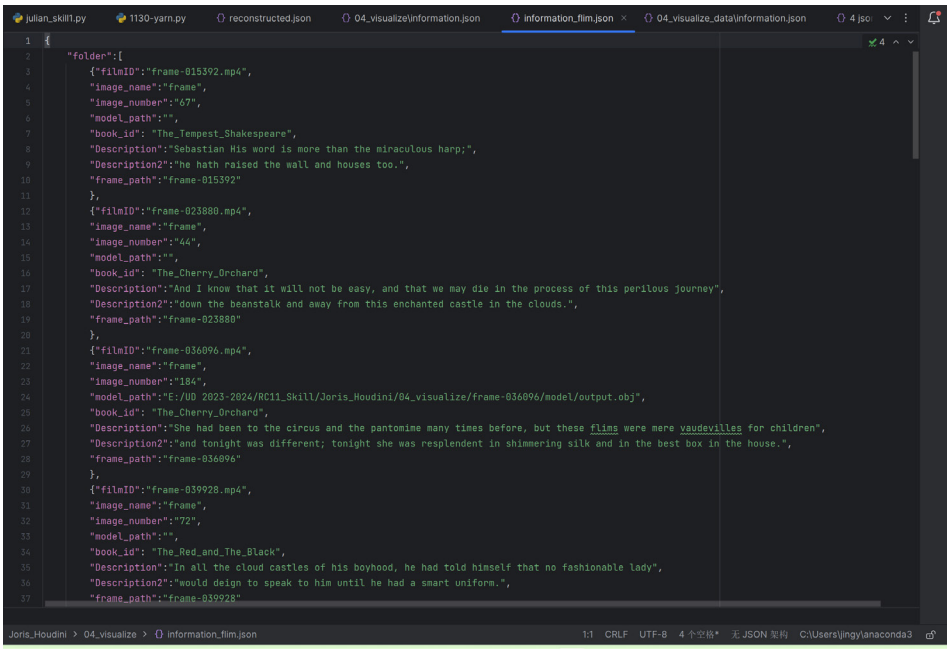
First Step

```
In [41]: video_folder = "E:\\UD 2023-2024\\RC11_Skill\\Joris_Houdini\\04_visualize\\film"
         image_root = "E:\\UD 2023-2024\\RC11_Skill\\Joris_Houdini\\04_visualize"

In [42]: videos = glob.glob(os.path.join(video_folder, '*.mp4'))

In [43]: videos

Out[43]: ['E:\\UD 2023-2024\\RC11_Skill\\Joris_Houdini\\04_visualize\\film\\frame-015392.mp4',
          'E:\\UD 2023-2024\\RC11_Skill\\Joris_Houdini\\04_visualize\\film\\frame-023880.mp4',
          'E:\\UD 2023-2024\\RC11_Skill\\Joris_Houdini\\04_visualize\\film\\frame-036096.mp4',
          'E:\\UD 2023-2024\\RC11_Skill\\Joris_Houdini\\04_visualize\\film\\frame-039928.mp4',
          'E:\\UD 2023-2024\\RC11_Skill\\Joris_Houdini\\04_visualize\\film\\frame-041073.mp4',
          'E:\\UD 2023-2024\\RC11_Skill\\Joris_Houdini\\04_visualize\\film\\frame-060592.mp4',
          'E:\\UD 2023-2024\\RC11_Skill\\Joris_Houdini\\04_visualize\\film\\frame-068543.mp4',
          'E:\\UD 2023-2024\\RC11_Skill\\Joris_Houdini\\04_visualize\\film\\frame-072800.mp4',
          'E:\\UD 2023-2024\\RC11_Skill\\Joris_Houdini\\04_visualize\\film\\frame-094787.mp4',
          'E:\\UD 2023-2024\\RC11_Skill\\Joris_Houdini\\04_visualize\\film\\frame-097974.mp4',
          'E:\\UD 2023-2024\\RC11_Skill\\Joris_Houdini\\04_visualize\\film\\frame-099805.mp4',
          'E:\\UD 2023-2024\\RC11_Skill\\Joris_Houdini\\04_visualize\\film\\frame-104435.mp4',
          'E:\\UD 2023-2024\\RC11_Skill\\Joris_Houdini\\04_visualize\\film\\frame-106288.mp4',
          'E:\\UD 2023-2024\\RC11_Skill\\Joris_Houdini\\04_visualize\\film\\frame-107096.mp4',
          'E:\\UD 2023-2024\\RC11_Skill\\Joris_Houdini\\04_visualize\\film\\frame-122726.mp4',
          'E:\\UD 2023-2024\\RC11_Skill\\Joris_Houdini\\04_visualize\\film\\frame-126240.mp4',
          'E:\\UD 2023-2024\\RC11_Skill\\Joris_Houdini\\04_visualize\\film\\frame-128919.mp4',
          'E:\\UD 2023-2024\\RC11_Skill\\Joris_Houdini\\04_visualize\\film\\frame-137520.mp4',
          'E:\\UD 2023-2024\\RC11_Skill\\Joris_Houdini\\04_visualize\\film\\frame-140072.mp4',
          'E:\\UD 2023-2024\\RC11_Skill\\Joris_Houdini\\04_visualize\\film\\frame-171595.mp4']
```

```
In [40]: for video_path in videos:
             cap = cv2.VideoCapture(video_path)

             video_name = os.path.basename(video_path).split('.')[0]

             save_folder = os.path.join(image_root, video_name)

             if not os.path.exists(save_folder):
                 os.makedirs(save_folder)

             frame_no = 0
             while cap.isOpened():
                 ret, frame = cap.read()
                 if not ret:
                     break

                 if frame_no % 10 == 0:
                     name = f"frame_{frame_no}.jpg"
                     target = os.path.join(save_folder, name)
                     cv2.imwrite(target, frame)

                 frame_no += 1

             cap.release()
```

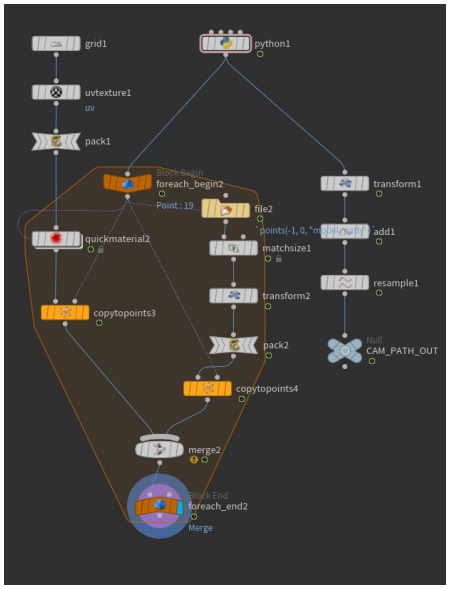
Using python code to extract the frames from the downloaded video and save them in different folders.

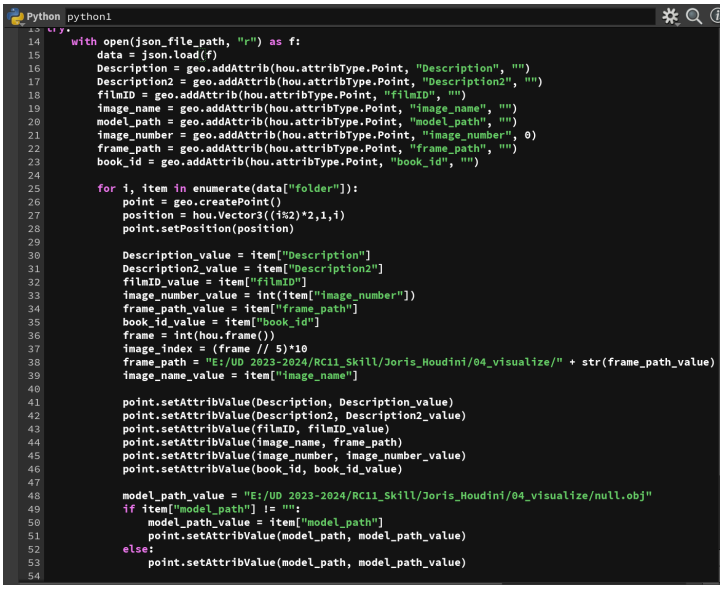
Second Step



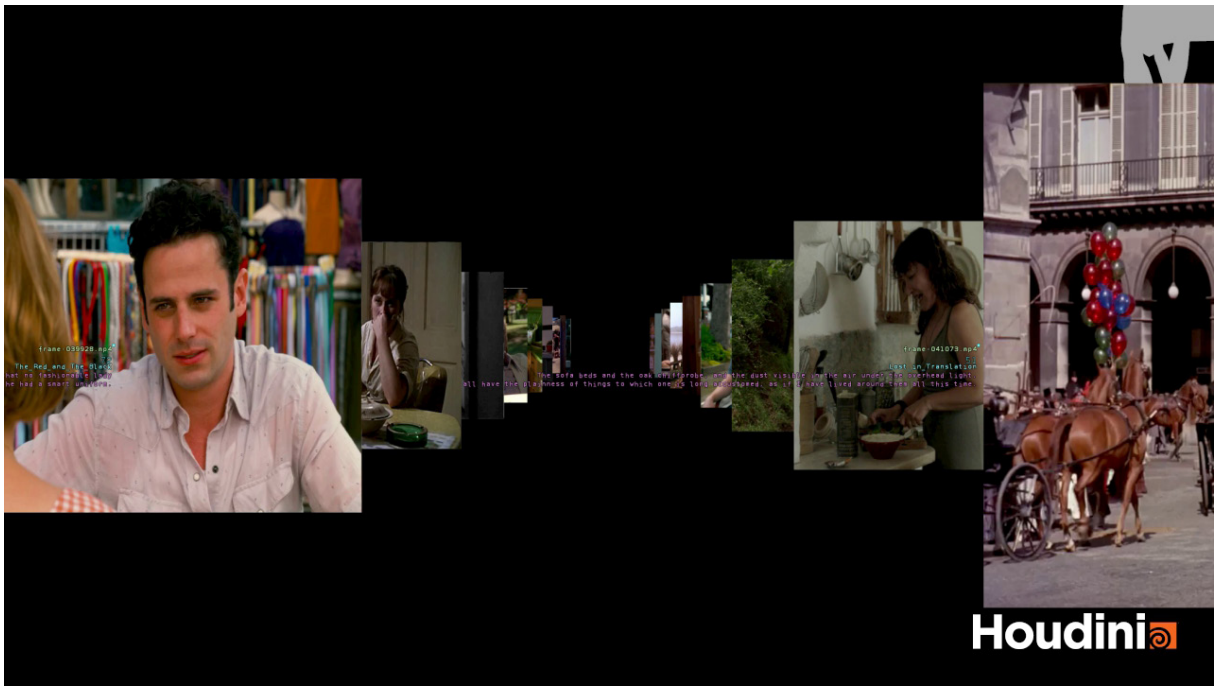
Build a json file with pycharm to import information from 20 films and corresponding literary texts.

Third Step





Import the json file, extract all the information from it, create a point for each value and attribute it, and assign the frames as texture to the grid created from the points. loading the model through model path and placing it to the corresponding position.


Houdini

FINAL_ASSIGNMENT_3

Tutor: Ceel Pierik

All the code parts are in this file:
https://github.com/UD-Skills-2023-24/github-codebook-RC11-23114486/blob/main/Final_Assignment_3_Ceel/SkillFinal_ceel.ipynb

Excercise 1

1. Implement this algorithm in Python. Use the NumPy ndarray object for your matrices

```
[106] import numpy as np
```

```
[121] def square_matrix_multiply(A, B):
          n = A.shape[0]
          C = np.zeros((n, n))

          for i in range(n):
              for j in range(n):
                  for k in range(n):
                      C[i][j] += A[i][k] * B[k][j]

          return C
```

```
[124] A = np.array([[1,5],[6,3]])
```

```
[125] B = np.array([[4,8],[7,2]])
```

```
[126] C = square_matrix_multiply(A, B)
```

```
[127] C

      array([[39., 18.],
             [45., 54.]])
```

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

The asymptotic time complexity is O(n^3).

outer loop "for i in range(n):" will execute n times,

middle loop "for j in range(n):" for each "i" it will execute n times, so it's n x n

inner loop "for k in range(n):" for each (i,j) it will execute n times, so it's n x n x n

So it's **O(n^3).**

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

```
[94] def square_matrix_multiply_2(A, B):
         n = len(A)
         C = [[0] * n for _ in range(n)]

         for i in range(n):
             for j in range(n):
                 for k in range(n):
                     C[i][j] += A[i][k] * B[k][j]

         return C
```

```
[88] A1 = np.random.randint(0, 100, size=(10, 10)).tolist()
     B1 = np.random.randint(0, 500, size=(10, 10)).tolist()
```

```
[89] A2 = np.random.rand(100, 100).tolist()
     B2 = np.random.rand(100, 100).tolist()
```

```
[95] start_time = time.time()
     C = square_matrix_multiply_2(A1, B1)
     end_time = time.time()

     cost_time = end_time - start_time
     print(cost_time)
```

```
0.0006127357482910156
```

```
[96] start_time = time.time()
     C = square_matrix_multiply_2(A2, B2)
     end_time = time.time()

     cost_time = end_time - start_time
     print(cost_time)
```

```
0.1893765926361084
```

4. Compare with built-in multiplication functions

```
[98] start_time = time.time()
     np.matmul(A1, B1)
     end_time = time.time()

     cost_time = end_time - start_time
     print(cost_time)
```

```
0.0003981590270996094
```

```
[99] start_time = time.time()
     np.matmul(A2, B2)
     end_time = time.time()

     cost_time = end_time - start_time
     print(cost_time)
```

```
0.013062238693237305
```

obviously the built-in function is more efficient.

5. Look at multiplying more than two matrices

```
[103] def matrices_multiply(matrices):

          final_matrix = matrices[0]

          for i in range(1, len(matrices)):
              final_matrix = square_matrix_multiply(final_matrix, matrices[i])

          return final_matrix
```

```
[101] A = np.array([[1,5],[6,3]])
      B = np.array([[4,8],[7,2]])
      C = np.array([[9,-7],[3,-3]])
```

```
[104] result = matrices_multiply([A, B, C])

      [[39. 18.]
       [45. 54.]]
      [[ 405. -327.]
       [ 567. -477.]]
```

Excercise 2

1.Describe and explain the algorithm.

Recursiveness: This algorithm keeps splitting the matrices into quarters by determining the size of each matrices until the matrices pnly has 1 elements. Then calculate the product of each 1 x 1 matrices, add them together to get the result of the previous level of calculation. Finally merge step by step to get the result.

Divide-and-conquer: This algorithm divid the complex problem of multiplying two large matrices to the base conquerable case of multiplication of two elements and addition between matrices.

2. Implement the recursive algorithm in Python.

Algorithm 2: Square-Matrix-Multiply-Recursive

SMMRec(A,B):
n = nr of rows of A
let C be a new $n \times n$ matrix
if $n == 1$ **then**
  $c_{11} = a_{11} \cdot b_{11}$
else
  quarter matrices A, B, and C
  $C_{11} = \text{SMMRec}(A_{11}, B_{11}) + \text{SMMRec}(A_{12}, B_{21})$
  $C_{12} = \text{SMMRec}(A_{11}, B_{12}) + \text{SMMRec}(A_{12}, B_{22})$
  $C_{21} = \text{SMMRec}(A_{21}, B_{11}) + \text{SMMRec}(A_{22}, B_{21})$
  $C_{22} = \text{SMMRec}(A_{21}, B_{12}) + \text{SMMRec}(A_{22}, B_{22})$
return C

These lines are straightforward to implement

These lines hide a lot of complexity

```
[ ] def split_matrix(A, n):
        A11 = [row[:n] for row in A[:n]]
        A12 = [row[n:] for row in A[:n]]
        A21 = [row[:n] for row in A[n:]]
        A22 = [row[n:] for row in A[n:]]
        return A11, A12, A21, A22
```

```
[ ] #Straightforward to implement
    def add2matrix(A, B):
        n = len(A)
        return [[A[i][j] + B[i][j] for j in range(n)] for i in range(n)]
```

```
[ ] #Straightforward to implement
    def combine_matrix(A, B, C, D):
        row_1 = [A[i] + B[i] for i in range(len(A))]
        row_2 = [C[i] + D[i] for i in range(len(C))]
        return row_1 + row_2
```

```
[ ] def Square_Matrix_Multiply_Recursive(A,B):
        n = len(A)
        C = [[0 for _ in range(n)] for _ in range(n)]

        if n == 1:
            return [[A[0][0] * B[0][0]]]

        else:
            A11, A12, A21, A22 = split_matrix(A, n//2)
            B11, B12, B21, B22 = split_matrix(B, n//2)

            #Hide a lot of complexity
            C11 = add2matrix(Square_Matrix_Multiply_Recursive(A11, B11), Square_Matrix_Multiply_Recursive(A12, B21))
            C12 = add2matrix(Square_Matrix_Multiply_Recursive(A11, B12), Square_Matrix_Multiply_Recursive(A12, B22))
            C21 = add2matrix(Square_Matrix_Multiply_Recursive(A21, B11), Square_Matrix_Multiply_Recursive(A22, B21))
            C22 = add2matrix(Square_Matrix_Multiply_Recursive(A21, B12), Square_Matrix_Multiply_Recursive(A22, B22))

            C = combine_matrix(C11, C12, C21, C22)

            return C
```

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.

1. For base case: $O(1)$
   - only need 1 multiplication.

2. For divide step: $O(1)$
   divide a $n \times n$ matrix into 4 $\frac{n}{2} \times \frac{n}{2}$ matrices,
   can be seen as constant time,

3. conquer step:
   each C_{ij} need 2 multiplication, so it's $8T(\frac{n}{2})$

4. combine step: $O(n^2)$
   for each C_{ij}, add 2 submatrices,
   for each submatrices, it has $\frac{n}{2} \times \frac{n}{2}$ elements,

$$T(n) = \begin{cases} O(1) & \text{if base case} \\ n^3 & \text{else} \end{cases}$$

$T(n) = 8T(\frac{n}{2}) + f(n)$

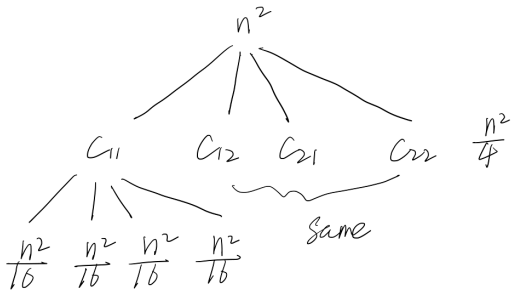
scale down the complex problem from n to $\frac{n}{2}$, and there are 8 submatrix multiplations.

f(n) are other works beside recursion. In before analysis, maximum complexity of other steps is n^2. So $f(n) = n^2$.

$T(n) = n^{\log_2 8} + n^2 = n^3 + n^2$.

So the com. asymptotic complexity is $O(n^3)$.

4. Do a tree analysis



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

```
[118] A2 = np.random.rand(256, 256)
      B2 = np.random.rand(256, 256)
```

```
[119] start_time = time.time()
      Square_Matrix_Multiply_Recursive(A2,B2)
      end_time = time.time()

      cost_time = end_time - start_time
      print(cost_time)
```

```
63.86083126068115
```

```
[128] start_time = time.time()
      square_matrix_multiply(A2, B2)
      end_time = time.time()

      cost_time = end_time - start_time
      print(cost_time)
```

```
17.688087940216064
```

Excercise 3

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

For this two algorithm, the aysmptotic complexity of addition/subtraction is O(n^2). And the aysmptotic complexity of the recursive matrix multiplication is O(n^3).
When dealing with very large matrices, the width of the recursion tree at the bottom is vast, making the computational cost extremely high, so reducing even a single multiplication operation is very important.

2. Do a complexity analysis of the Strassen algorithm.

$$T(n) = \begin{cases} O(1) & \text{if base case} \\ n^{\log_2 7} \end{cases}$$

$$T(n) = 7T\left(\frac{n}{2}\right) + f(n)$$

1. for $f(n)$:

Through the pseudo code, there 18 addition and subtration in addition to the recursion part, for each time, it will calculate $\left(\frac{n}{2}\right)^2 = \frac{n^2}{4}$ elements. (elements in each $\frac{n}{2} \times \frac{n}{2}$ submatrices).
So $f(n) = O(n^2)$

2. for $7T\left(\frac{n}{2}\right)$

$a=7,\ b=2,\ n^{\log_b a} = n^{\log_2 7} > n^2$

So the overall complexity is $O(n^3)$.

3. Implement and test the algorithm

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

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

     def combine_matrix(A, B, C, D):
         First_row = [A[i] + B[i] for i in range(len(A))]
         Second_row = [C[i] + D[i] for i in range(len(C))]
         return First_row + Second_row
```

```
[ ]  def strassen(A, B):

         n = len(A)
         if n == 1:
             return [[A[0][0] * B[0][0]]]

         else:
             A11, A12, A21, A22 = split_matrix(A, n//2)
             B11, B12, B21, B22 = split_matrix(B, n//2)

             P1 = strassen(matrix_add(A11, A22), matrix_add(B11, B22))
             P2 = strassen(matrix_add(A21, A22), B11)
             P3 = strassen(A11, matrix_sub(B12, B22))
             P4 = strassen(A22, matrix_sub(B21, B11))
             P5 = strassen(matrix_add(A11, A12), B22)
             P6 = strassen(matrix_sub(A21, A11), matrix_add(B11, B12))
             P7 = strassen(matrix_sub(A12, A22), matrix_add(B21, B22))

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

             C = combine_matrix(C11, C12, C21, C22)
             return C
```