Multimedia Web Database

Multimedia Gesture Recognition

Group 6 Members

- Cole Ruter 1211475262
- Joshua Marriott 1212700656
- Neel Jepaliya 1212689619
- Pinaki Saha 1218486737
- **Sheen Dullu 1217167770**
- Venkat Charan Guduru 1216985133

Abstract

This phase of the project is based on understanding and implementing the vector models for multi-variate time-series data sets and to extract information such as building common features and finding the similarity and how to process those features. The project uses the time-series dataset and walks from quantizing the dataset, extracting the common features, building the heatmap, and finding similar gestures. Through this project, I was able to experiment on the number of features to use, importance of structuring the data and how to build an optimized system.

Keywords: Vector Models, Features, Term Frequency, Term Frequency- Inverse Document Frequency, window length, Gaussian Bands

Introduction

As part of the course, CSE 515 Multimedia and Web Databases, this phase of the project is based on understanding the process of working with multimedia and continuous data and finding similarities among the multimedia objects. Using multi-variate time-series data for Z-dimension of various gestures, the product of the algorithm is presenting the top 10 similar gestures for the given gesture.

Terminology

TF: Abbreviate for Term Frequency. TF presents the weight/frequency of a word or object in a document. Hence, TF presents the description power of a word for the document.

IDF: Abbreviate for Inverse Document Frequency. IDF presents how unique a word is in collection of documents, hence providing the discriminatory factor.

TF-IDF: Abbreviate for Term Frequency-Inverse Document Frequency. TF-IDF is the product of TF and IDF to provide the importance of a word in a document given in a collection of documents or corpus.

TF-IDF2: As the object in the project consists of 20 sensor values, TF-IDF2 is used to find the importance of word across all sensors in a file.

Window Length: Window length is the number of the sequential data points which would be extracted to create a word.

Shift Space: Shift space is the unit of time one must skip.

Normalize: To normalize the data means to rescale the numeric data/attributes within a given range. This helps to make features consistent with each other. In this case, the range to normalize is [-1, 1].

Quantization: To quantize means to constraint the continuous values to a discrete set. In this case, assigning the continuous time-series data to fixed band values from 1 to 2*resolution

Goal Description

The goal of the project is to thoroughly understand the process of determining similar objects from the database. Using the time-series dataset and walks from quantizing the dataset, extracting the common features, building the heatmap, and finding similar gestures. Through this project, I was able to experiment on the number of features I should be using, considering the increase in the cost by increasing the number of features.

Assumptions

- The directory *dir* for the data will have .csv files with multi-variate time-series data.
- The directory *dir* will be the location along with folder name. For example, the Folder *Z* has all the gesture files, so *dir* will be

  <*folder_path…\Z*>.

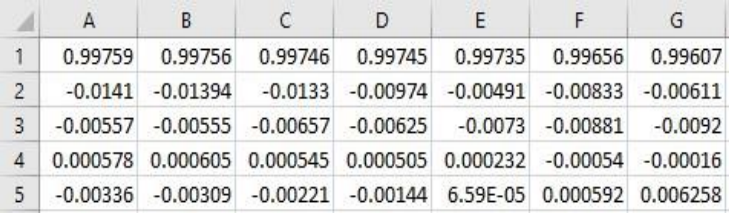
In my case, *dir* input was

D:\ASU\Courses\MWDB\Project\Code\Z

Description

Dataset

The data used during the project is the multi-variate time series data of the Z-Dimensions based on the gestures performed by human beings with 20 sensors attached across their body. Below is the screenshot of the data.



	A	B	C	D	E	F	G
1	0.99759	0.99756	0.99746	0.99745	0.99735	0.99656	0.99607
2	-0.0141	-0.01394	-0.0133	-0.00974	-0.00491	-0.00833	-0.00611
3	-0.00557	-0.00555	-0.00657	-0.00625	-0.0073	-0.00881	-0.0092
4	0.000578	0.000605	0.000545	0.000505	0.000232	-0.00054	-0.00016
5	-0.00336	-0.00309	-0.00221	-0.00144	6.59E-05	0.000592	0.006258

Figure 1. Snapshot of the Z-Dimension dataset

Tasks

This phase is divided into 4 well-defined tasks that advance towards reaching our goal. Below are the tasks as well as how I designed my algorithm to perform all these tasks.

Task 1

This task focuses on normalizing and quantizing the multi-variate time-series data to build a gesture dictionary.

User Input-

- File path to the directory (dir)
- Window Length for length of words w
- Resolution r
- Shift Length s

Process-

- I store the directory value in the file "parameter.txt", so that I do not ask for code directory again.
- I created Gaussian Bands by dividing the range [-1, 1] using the user input resolution r and stored these values to further use for quantization. To determine the length of the bands, I first used the given formula with parameters, mean μ = 0 and standard deviation σ = 0.25.

$$length_i = 2 \times \frac{\int_{(i-r-1)/r}^{(i-r)/r} Gaussian_{(\mu=0.0,\sigma=0.25)}(x) \;\; \delta x}{\int_{-1}^{1} Gaussian_{(\mu=0.0,\sigma=0.25)}(x) \;\; \delta x}$$

For my algorithm, the bands are:

- Band 1: [1, 0.9924]
- Band 2: (0.99245, 0.81763]
- Band 3: (0.81763, 0.0]
- Band 4: (0.0, -0.81763]
- Band 5: (-0.81763, -0.99245]
- Band 6: (-0.99245, -1.0]

- Then I collected all the files and normalized their values between [-1, 1] with respect to each sensor using the given formula,

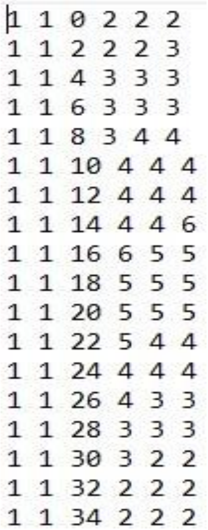
$$x''' = (b - a)\frac{x - \min x}{\max x - \min x} + a$$

where,

- x = value of a unit/cell in sensor
- min x = minimum value for a sensor where x appears
- max x = maximum value for a senor where x appears
- [a, b] = [-1, 1]
- x^m = normalized value for sensor x

- In the next step, I quantized each value to the Gaussian bands created above. Then, for each sensor, I used window length, and shift space from the user input to captured the words within the window length in a sequence of their time units and using shift space I moved my window accordingly and stored these values as words in the following format for each file separately.

< filename sensor_id window_start_time <word>>



```
1 1 0 2 2 2
1 1 2 2 2 3
1 1 4 3 3 3
1 1 6 3 3 3
1 1 8 3 4 4
1 1 10 4 4 4
1 1 12 4 4 4
1 1 14 4 4 6
1 1 16 6 5 5
1 1 18 5 5 5
1 1 20 5 5 5
1 1 22 5 4 4
1 1 24 4 4 4
1 1 26 4 3 3
1 1 28 3 3 3
1 1 30 3 2 2
1 1 32 2 2 2
1 1 34 2 2 2
```

Figure 2. Snapshot of 1.wrd

Task 2

This task focuses on creating gesture vectors based on TF, TF-IDF, and TF-IDF2.

User Input -

- File path to the directory

Process-

- Using all the output files from Task 1, I created a dictionary *all_words* of all the words that occurred in at least once in a sensor for every file. I did so to keep track of all the features(sequence of words) that are used in the sensor instead of using all possible combinations of words for the features which would lead to unnecessary cost. Such as
    - For the given user input of *resolution* as 3 and *window length* as 3, the total number of possible words become $(2*3)^3$ = 216, whereas the total number of words that appeared at least once in the test data is 144. This would include unnecessary computation cost.
    - Also, if we increase the window length to 4, then the total number of possible words will become 1296.
- For each *.wrd* file, I used *all_words* dictionary to assign each sensor data with it and capture the frequency of a word in that sensor otherwise keep it as 0.
- Then using the output data from the above step, I perform calculations for TF, TF-IDF, and TF-IDF2 using the below formulas:

    - For TF,

    As 20 sensors are placed on different parts of the body, I chose to calculate TF for each word within a sensor individually.

    For each sensor in each file, TF = n/K

    where,

    ○ n = count of a word in a sensor

    ○ K = total words present in that sensor

    - For TF-IDF,

    I calculated TF-IDF for each word in the specific sensor with respect to all file's directory wide.

    TF-IDF for a word in a file to a sensor directory-wide = TF of a word in file * log(N/m)

    where,

    ○ N = total number of files

    ○ m = number of files that contain at least one of the given words in a sensor

    - For TF-IDF2,

    I calculated TF-IDF for each word in a file across all other sensors in that file.

    TF-IDF2 for a word in a file across all its sensors = TF of a word * log(N/m)

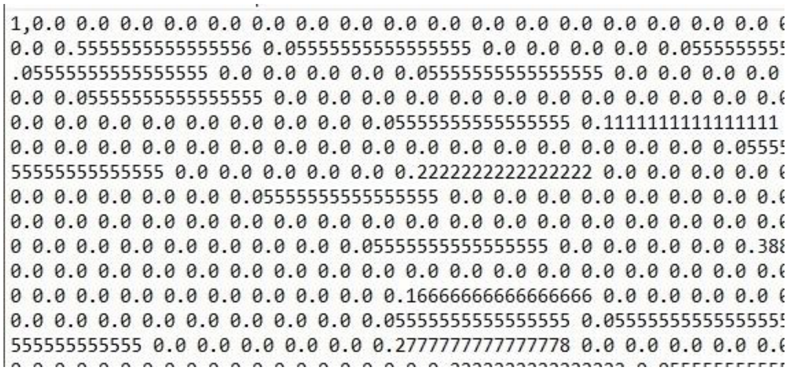
    where,

- N = total number of sensors
- m = number of files that contain at least one of the given words in its sensor

To store these values in a file, I used the word dictionary *all_words* to keep track of the position of the word in a sensor and appended all the sensor vectors simultaneously with keeping track of the position of the word in the vector. Then store it in *vectors.txt* in the same directory.

I saved the output in the file using the following structure:

<filename <TF for 20 sensors> <TF-IDF for 20 sensors> <TF-IDF2 for 20 sensors>



```
1,0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 
0.0 0.5555555555555556 0.05555555555555555 0.0 0.0 0.0 0.0 0.055555555
.05555555555555555 0.0 0.0 0.0 0.0 0.05555555555555555 0.0 0.0 0.0 0.0 
0.0 0.05555555555555555 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.
0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.05555555555555555 0.1111111111111111 
0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0555
55555555555555 0.0 0.0 0.0 0.0 0.0 0.2222222222222222 0.0 0.0 0.0 0.0 
0.0 0.0 0.0 0.0 0.0 0.05555555555555555 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.
0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.
0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.05555555555555555 0.0 0.0 0.0 0.0 0.38
0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.
0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.16666666666666666 0.0 0.0 0.0 0.0 
0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.05555555555555555 0.0555555555555555
555555555555 0.0 0.0 0.0 0.0 0.0 0.2777777777777778 0.0 0.0 0.0 0.0 0.
```

Figure 3. Snapshot of vectors.txt

Task 3

This task focuses on creating Heatmap for a file based on TF, TF-IDF, and TF-IDF2.

User Input-

- File with the file path
- Selection from TF, TF-IDF, TF-IDF2

Process-

To construct the heatmap using time and sensors, I used the *vectors.txt* file and the respective *.wrd* file to load all the TF, TF-IDF, and TF-IDF2 values and to get the time information for the words in that file. Since I stored these vectors according to the position of the words, I was able to retrieve the word information along with sensors id. Through *.wrd* file I was able to retrieve time information with sensor id as well as words. I was able to construct a sensor id vs time graph with cell values containing words, which I replaced by their respective TF, TF-IDF, and TF-IDF2 values from the *vectors.txt* file. Then plotting and displaying Heatmap for TF, TF-IDF, and TF-IDF2.

For the Heatmap, the x-axis contains the entries for the duration of time-series data and the y-axis contains the sensor id.

Below are the heatmap plots for the test1.csv file based on TF, TF-IDF, and TF-IDF2.

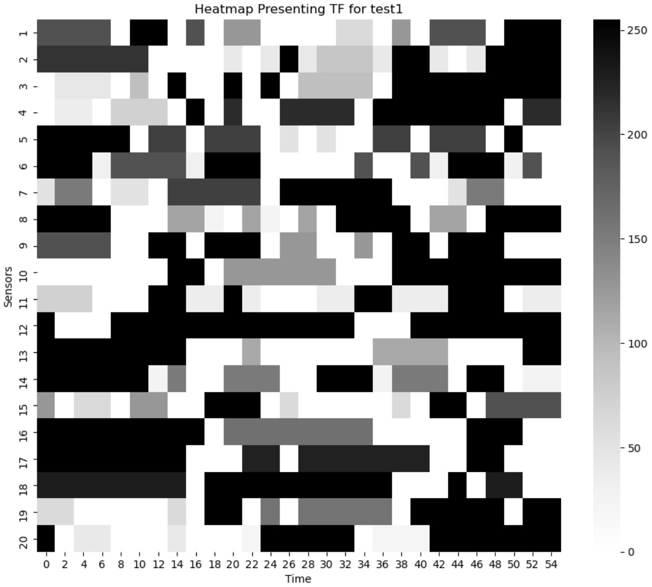


Figure 4. Snapshot of Test1.csv Heatmap based on TF

The above Figure 5 is the heatmap for test1.csv from the given test data. If the area on the plot is dark, then it means that the TF value of a word within a sensor is high that appeared at that time. Meaning that part of the gesture is repeated multiple times. If the area of the plot is white, it means that the word at that time is appeared only once in the sensor, hence leading to lower TF value.

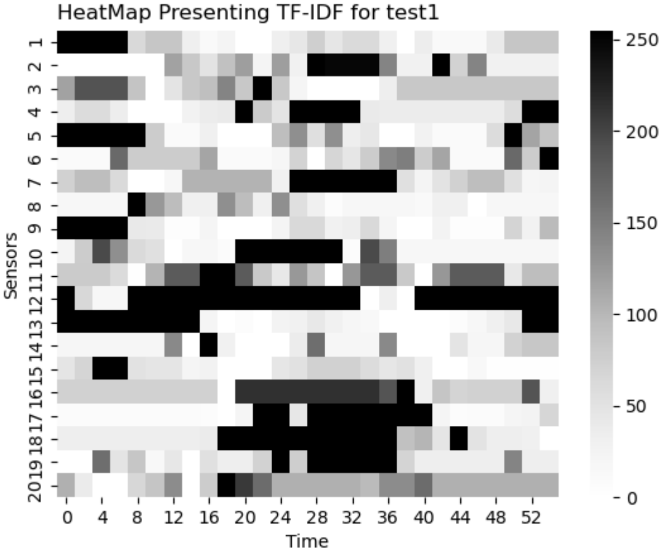


Figure 5. Snapshot of Test1.csv Heatmap based on TF-IDF

The above Figure 6 is the heatmap for test1.csv from the given test data. If the area on the plot is dark, then it means that the TF-IDF value of a word, that appeared at that time, for a sensor is high. Meaning that part of the gesture is rarely seen across all the files/gestures in the directory. If the area of the plot is white, it means that the word at that time is appeared in all the files/gestures in the directory, hence leading to lower TF-IDF value.

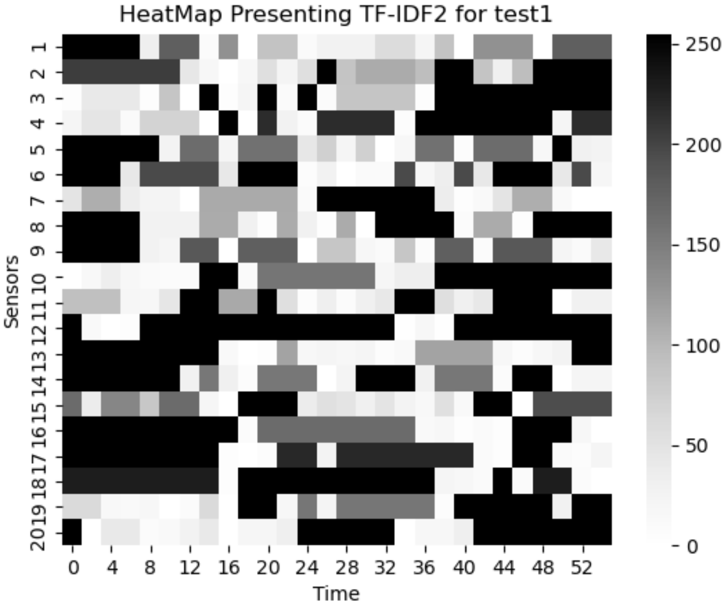


Figure 6. Snapshot of Test1 Heatmap based on TF-IDF2

The above Figure 7 is the heatmap for test1.csv from the given test data. If the area on the plot is dark, then it means that the TF-IDF2 value of a word, that appeared at that time, for a sensor is high. Meaning that part of the gesture is rarely seen across all the sensors in that file. If the area of the plot is white, it means that the word at that time is appeared in all the sensors in that file, hence leading to lower TF-IDF2 value.

Task 4

This task focuses on finding the 10 similar gestures to a file

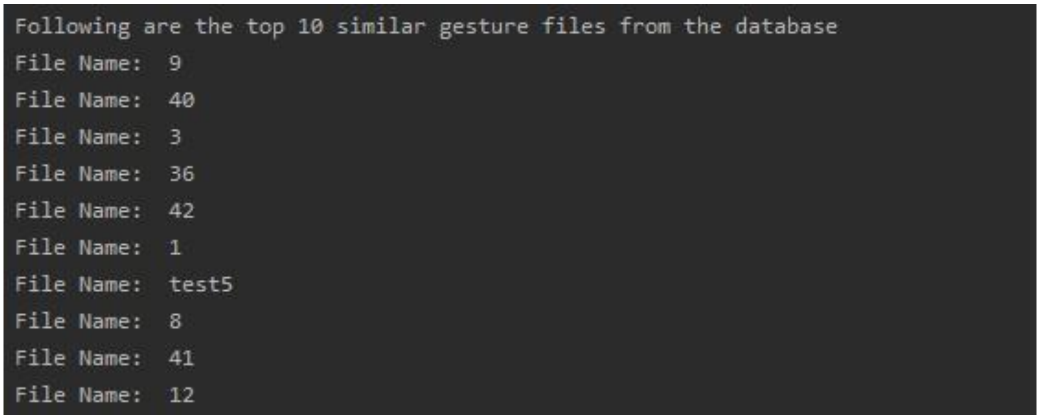
User Input-

- File with the file path
- Selection from TF/TF-IDF/TF-IDF2

Process-

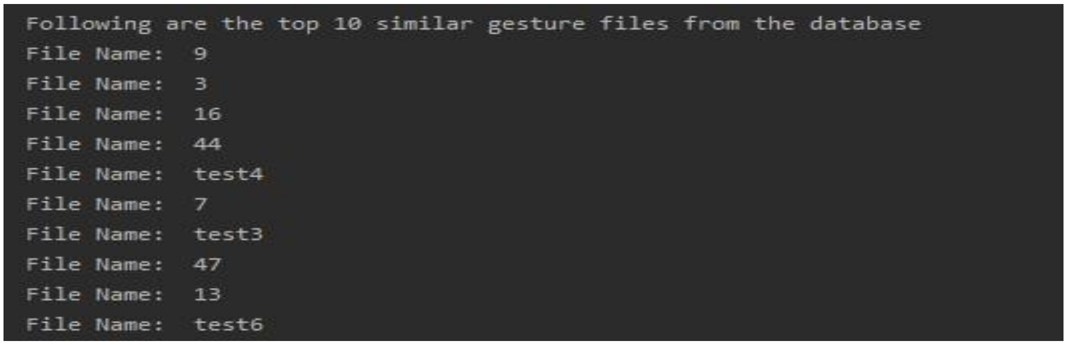
I used the *vectors.txt* file to load all the TF, TF-IDF, and TF-IDF2 values for every file. Using the user input, I then calculated the Euclidean distance of all the other files with respect to the user input file. Through Euclidean distance, I get the distance between the gestures. For calculating the similarity, if the Euclidean distance between the gesture is low, it means those 2 gestures are similar. I only print the top 10 closest gestures part from the file itself.

Below are the snapshots for the test1.csv file based on TF, TF-IDF, and TF-IDF2.



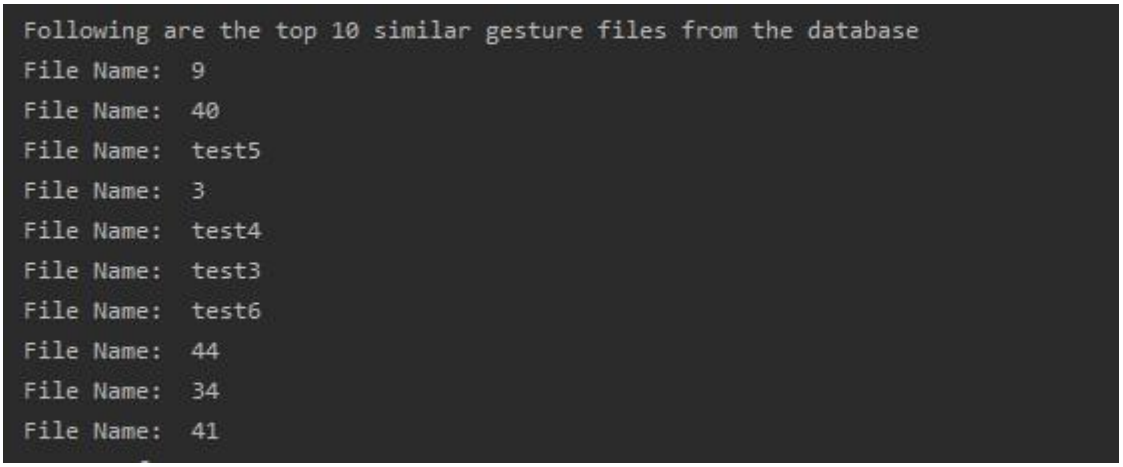
```
Following are the top 10 similar gesture files from the database
File Name:  9
File Name:  40
File Name:  3
File Name:  36
File Name:  42
File Name:  1
File Name:  test5
File Name:  8
File Name:  41
File Name:  12
```

Figure 7. Snapshot of Top 10 similar gestures(apart from itself) for Test1.csv based on TF



```
Following are the top 10 similar gesture files from the database
File Name:  9
File Name:  3
File Name:  16
File Name:  44
File Name:  test4
File Name:  7
File Name:  test3
File Name:  47
File Name:  13
File Name:  test6
```

Figure 8. Snapshot of Top 10 similar gestures(apart from itself) for Test1.csv based on TF-IDF

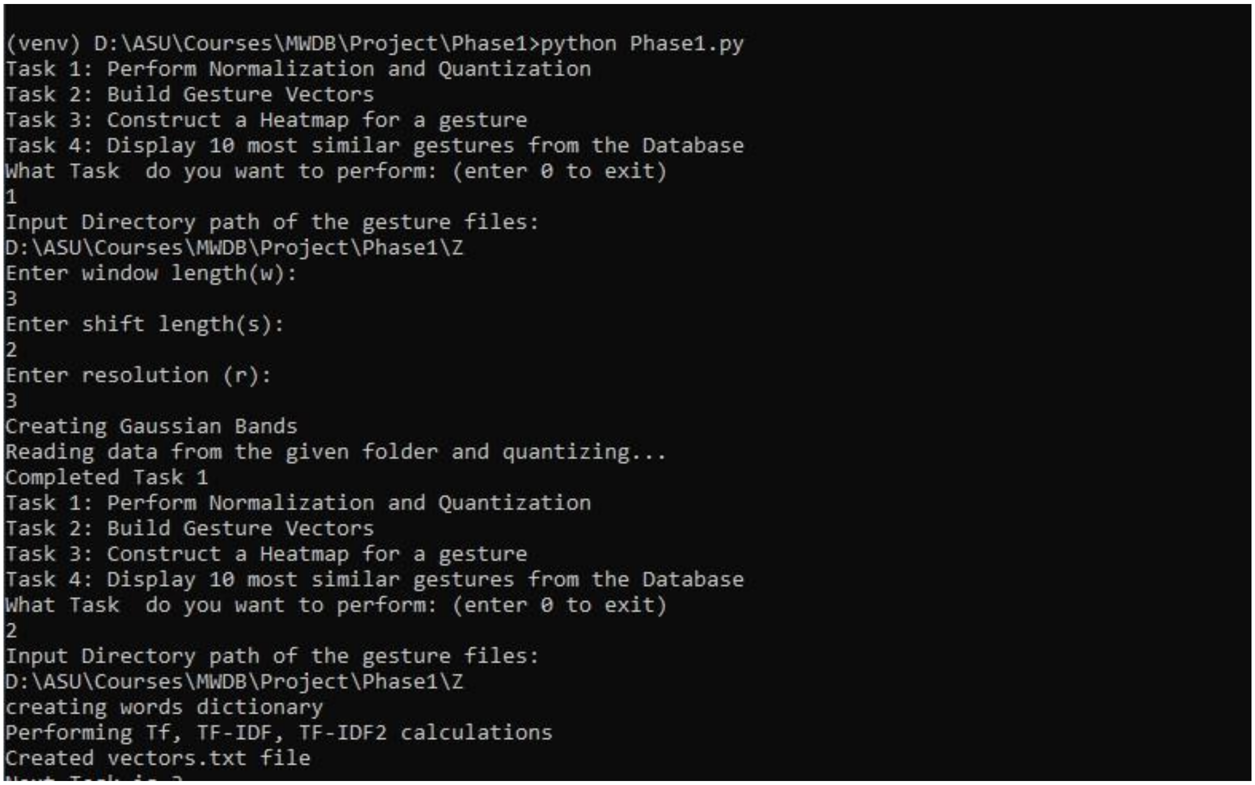


```
Following are the top 10 similar gesture files from the database
File Name:  9
File Name:  40
File Name:  test5
File Name:  3
File Name:  test4
File Name:  test3
File Name:  test6
File Name:  44
File Name:  34
File Name:  41
```

Figure 9. Snapshot of Top 10 similar gestures(apart from itself) for Test1.csv based on TF-IDF2

Interface specification

The system interface for my algorithm is based on user input methods.



```
(venv) D:\ASU\Courses\MWDB\Project\Phase1>python Phase1.py
Task 1: Perform Normalization and Quantization
Task 2: Build Gesture Vectors
Task 3: Construct a Heatmap for a gesture
Task 4: Display 10 most similar gestures from the Database
What Task  do you want to perform: (enter 0 to exit)
1
Input Directory path of the gesture files:
D:\ASU\Courses\MWDB\Project\Phase1\Z
Enter window length(w):
3
Enter shift length(s):
2
Enter resolution (r):
3
Creating Gaussian Bands
Reading data from the given folder and quantizing...
Completed Task 1
Task 1: Perform Normalization and Quantization
Task 2: Build Gesture Vectors
Task 3: Construct a Heatmap for a gesture
Task 4: Display 10 most similar gestures from the Database
What Task  do you want to perform: (enter 0 to exit)
2
Input Directory path of the gesture files:
D:\ASU\Courses\MWDB\Project\Phase1\Z
creating words dictionary
Performing Tf, TF-IDF, TF-IDF2 calculations
Created vectors.txt file
```

Figure 10. Snapshot of the interface

I designed the application according to the mentioned inputs and the flow for the system. The application once started will allow the user to perform the tasks on the go. The system will work accordingly even if the user performs Task 1 and exit the program and rerun it again from Task 2.

System Requirements/Installation

The System should be installed with Python 3.7.7.

Execution

To Run the System:

1. Unzip the file.
2. Open the folder **Code** in the command prompt
3. Create virtual environment folder: **python -m venv venv**
4. Activate the virtual environment: **venv\Scripts\activate**
5. Install all the packages required: **pip install -r requirements.txt**
6. Run the program file: **python Phase1.py**

7. Follow along the interface.

Related Word

This project is similar to *Human gesture recognition through a Kinect sensor*[1]. But we used a simple approach in determining the similarity among gestures from multi-variate time-series data.

Conclusion

Through this project I was able to experiment and learn the process of quantizing and creating features based on the requirements of the multimedia data. Also, learnt the importance of the number of features to use in multimedia retrieval system.

Bibliography

1. Ye Gu, Ha Do, Yongsheng Ou, Weihua Sheng, “Human gesture recognition through a Kinect sensor” 2012 IEEE International Conference on Robotics and Biomimetics (ROBIO), Guangzhou, China, Dec. 2012, pp: 1379 – 1384, doi: 10.1109/ROBIO.2012.6491161