Spring 6-9-2022

Determining American Sign Language Joint Trajectory Similarity Using Dynamic Time Warping (DTW)

Rohith Mandavilli
Dartmouth College, Rohith.Mandavilli.22@Dartmouth.edu

Determining American Sign Language Joint Trajectory Similarity Using Dynamic Time Warping (DTW)

An Honors Thesis Submitted to the Faculty in partial fulfillment of the requirements for the degree of Bachelor of Arts in Computer Science

Rohith Mandavilli

June 2, 2022

DARTMOUTH COLLEGE

Hanover, New Hampshire

Advised by Professor David Kraemer and Professor Devin Balckom

Abstract

As American Sign Language (ASL), the language used by Deaf/Hard of Hearing (D/HH) Americans has grown in popularity in recent years, an unprecedented number of schools and organizations now offer ASL classes. Many hold misconceptions about ASL, assuming it is easily learned; however due to its rich, complex grammatical construction, it's not mastered easily beyond a basic level. Therefore, it becomes ever more important to improve upon existing techniques to teach ASL. The Dartmouth Applied Learning Initiative (DALI) at Dartmouth college in coordination with the Robotics and Reality Lab developed an application on the Oculus Quest that helps D/HH individuals improve their ASL literacy. Though the app accurately predicts whether a user is signing letters correctly, it cannot verify signed words effectively due to the complexity involved in tracking motion and rotations. As a result this paper analyzes the effectiveness of using Dynamic Time Warping (DTW), a popular motion similarity comparison technique, to compare user-signed joint trajectories. I compute an 84% accuracy rate as a low bound for my algorithm due to factors involved in this calculation. This is primarily driven by one of the signs being imperfectly signed, and when we exclude that sign from analysis, our accuracy rate jumps to 92%. Therefore, I've identified a successful metric for validating the correctness of a signed word.

Preface

I would like to thank both my advisors Professor David Kraemer and Professor Devin Balckom for agreeing to oversee my project. Additionally, I immensely appreciate PhD student Julien Blanchet for constant guidance throughout the process.

I want to shoutout Ed the custodian from Raven House for entertaining conversation late at night while I was grinding out my honors thesis.

I am grateful to the DALI Lab for teaching me the fundamentals of software engineering. We have as a team built an application with immense potential to help people.

Lastly, thank you to my family, Satish, Vidhya, Isha and Bharghav Mandavilli for always wishing me the best and providing endless support in life.

Contents

1 Introduction

This section provides as an introduction to the motivation behind building an application. Specifically, the teach-student communication gap and difficulty of learning ASL motivate building an application that can provide consistent and objective feedback for learning signed words.

1.1 Teacher-Student Communication Gap

The relationship between teacher and student can be understood as a process of communication or a transfer of knowledge. This relationship is the crux of pedagogy and guides the growth of students. Previous research has argued that there exists an unrepresentable, transformative, communicative gap that cannot be closed nor controlled by either partner in question. Therefore, educational relations are never direct [2]. Ultimately, this implies that the student can never fully internalize verbal feedback given by their teacher because information is sometimes forgotten, applied incorrectly, or misunderstood.

To illustrate this problem, consider a swimmer and their coach. During practice, the coach times the swimmer and then provides feedback on how to lower the time, thus improving the run. From the perspective of the swimmer, receiving verbal feedback is obviously helpful, but hard to implement in practice. The coach may describe a few things for the swimmer to improve, like starting their turn earlier, increasing the length of their stroke, or diving deeper off the podium. Over many runs, there may be multiple pieces of feedback the swimmer has to remember, which can become cumbersome and tedious, potentially even frustrating.

This example obviously extends itself towards learning languages. To learn a new language, students must at least memorize words, learn pronunciation tendencies, and internalize a new grammatical system. Teachers generally agree that critical language awareness, interpretive skills, and historical consciousness are also required to truly understand a language [3]. Though ASL is not spoken, similar challenges such as overcoming cultural inhibitions and learning to interact with D/HH people are obstacles towards fluency [4].

In the context of a classroom setting, this gap has implications for ASL literacy. You can imagine a teacher asking a student to change the position of their hand as well as the rotation of a few of their joints simultaneously. They could also ask the user to mimic their facial expression or increase the speed at which they sign. The purpose of this application is to partially bridge this gap by providing a virtual environment where users can sign words and receive consistently correct feedback while minimizing effort required by an instructor.

1.2 Difficulty of Learning ASL

Hearing learners of ASL are forced to operate in a visual-gestural modality rather than the familiar aural/oral modality. There is no understood schema for how to interpret linguistic information visually. This can easily make the process of learning ASL mentally demanding. Some research has found that the translucency of a sign, the perceived relationship between a sign and its referent, and rated concreteness of the sign referent are significant predictors of individuals learning that specific sign [5]. Studying the meaning of a sign simultaneously with handshape and articulation has a significant impact on students remembering that sign.

Learners are also forced to become comfortable with explicitly using their body as forms of expression and communication. Interestingly, earlier work has found that students with theatrical talent are likely better prepared for non-oral communication [4]. These types of students are less likely to be afraid of making mistakes, so they don't experience pressure in the process of learning.

Both students and teachers generally perceive student ASL syntax and grammar to be their biggest weaknesses [4], immediately followed by insufficient repetition and practice in class. Though the scope of this app is therefore limited as it cannot teach grammar, the literacy gains made from practicing with it will smooth the transition to learning grammar as students need not focus on correctly signing words. Additionally, students report difficulty with perceiving and articulating hand signs accurately in terms of handshape and movement [4]. As discussed earlier,

the gap inherent to the teacher-student relationship makes correcting those errors arduous.

Informal language learning can occur outside classrooms through social interactions with native speakers or exposure to culture [7]. Indeed, students often study abroad in attempts to inculcate themselves with a specific language. However, research suggests that when ASL learners interact with members of the D/HH community, they are often not confident or reluctant [4].

As a result of these challenges, teachers of ASL traditionally maintain low expectations for their students. When new material is introduced, classroom pace slows and prior material is referenced to provide context [6]. This hampers students' learning ability and slows down the process of fluency. Therefore, it remains important to explore new techniques to teach sign language resilient to these obstacles.

1.3 ASL and Technology

The intention of this paper is to create a technological solution that can effectively teach individuals words in ASL, aiding student literacy. Eventually, I want to build this software into an application that can teach ASL in its entirety, including grammar. I will start with describing design principles related to successful software applications, then end with a discussion of the game mode I created.

1.3.1 The Effectiveness of Using Technology to Teach ASL

To start, Computer Assisted Instruction (CAI) has the potential to assist with early foreign language instruction [8]. In a study comparing the effectiveness of using technology in place of social interaction as an informal language learning setting, groups exposed to audiovisual mass media performed better on a curated speaking test than groups exposed to social interaction indicating exposure to technology promotes speaking proficiency [7].

To be clear, ASL should be postulated as a form of discourse far

more than varying degrees of knowledge of ASL. However, we can conceive of creating technology to assist with sign language literacy which would ultimately supplement classroom instruction [11]. Researchers in Brazil succeeded in creating a virtual environment allowing real-time communication via an online chat room. The experience improved the teaching and learning process and was readily accepted by learners [12].

1.3.2 Software Design Considerations

To be successful, in the early, stages software developers must target a specific lesson structure with pedagogical validity [9]. Furthermore, the ability of software to contribute pedagogical value is related to four central questions [10]:

1. Is increased technological sophistication related to increased effectiveness?

2. What attributes of the new technology can be profitably exploited for pedagogical purposes?

3. How can new technologies be successfully integrated into the curriculum?

4. Do new technologies provide for an efficient use of human and material resources?

I have found evidence that technology can positively impact the learning process of ASL, and have isolated specific questions that we need to address to ensure the effectiveness of the technology. The next section will detail the created game mode we've worked on to teach people ASL words.

1.3.3 Sign Language Virtual Reality Word Signing

Between the glossary mode teaching users letters and the fingerspelling game inviting users to sign each individual letter in a word to complete the word, I have developed a mode asking users to sign a given word displayed on a blackboard. The words are read in from a .txt file and the

user is given 4 seconds to sign the word. After the time ends, the avatar signs the motion the user just performed back to them, and then signs the given word correctly if the user was wrong. That way, the user can see what they did and immediately examine the differences between that and the correct sign. We believe that this somewhat models in person instruction. In a classroom, students will watch a teacher sign a word and attempt to copy them. After receiving feedback, hopefully they improve and eventually sign it correctly. Our application immediately provides an unbiased assessment of the quality of a hand sign and its mistakes, arguably more effective than a teacher who may have to help many students. However, as we have previously learned, translucency (understanding the meaning of a sign) is a major predictor of students being able to learn sign language. One consideration for the future would be some sort of implementation that conveys the meaning of a given word while asking the user to sign it.

I argue that this application resolves many of the difficulties involved in learning ASL. One issue that learners must face is adapting to the visual-gestural modality including being comfortable with using their body as expression. This application provides a break from reality where users are not worried about the perceived social pressure of signing because with the Oculus headset on, they will not be able to see anything besides the virtual environment. Additionally, since it has been agreed that increased repetition and practice would greatly benefit learners, the application already has demonstrated need. Because the feedback is provided by an algorithm, students could use the app outside of class without the need for a teacher. This saves time in the classroom and allows students to control their practice since they can choose to stop whenever they feel comfortable. Lastly, I believe it provides students with an avenue to get over nerves interacting with people from the D/HH community. Since learners hone their skills in a pressure-free environment, it will hopefully simultaneously build the confidence required to actually use sign language in the real world.

The application also fulfills the software design considerations laid out in the earlier subsection. Technological sophistication with the application can take the form of a multitude of things. Firstly, increased algorithmic performance smooths the movements and transitions of the

avatar, reducing lag. Smoothing makes avatar signs seamless and fluid, truly mimicking how D/HH people use ASL. Additionally, implementation of machine learning techniques could allow us to identify minute sections of a user sign that are wildly incorrect, allowing the application to give specific and pointed feedback to the user. Training data will be gathered from users over time, slowly building a model that can replace the feedback of a teacher.

There are a few attributes of the project that can be profitably exploited for classroom benefit. The most obvious example is supplementing the effort of the teacher with software. Over time, this will save money on behalf of schools and organizations by giving 1 on 1 instructions to students outside of class. Additionally, the data collected from the application will be useful in the future to find which specific signs students have trouble with and create a model that can ASL explain grammar and syntax.

For the third question, we can easily imagine the application to amplify in person curriculum through outside of class practice assuming students have or are provided with the resources to obtain an Oculus quest. Lastly, we can be sure that the software is an efficient use of resources because it's developed by college students learning about software engineering. The only consideration here is the upfront cost of implementing this technique in schools, but if the application were to have pedagogical value agreed upon by ASL experts and teachers, then schools will be incentivized to implement the application wherever necessary.

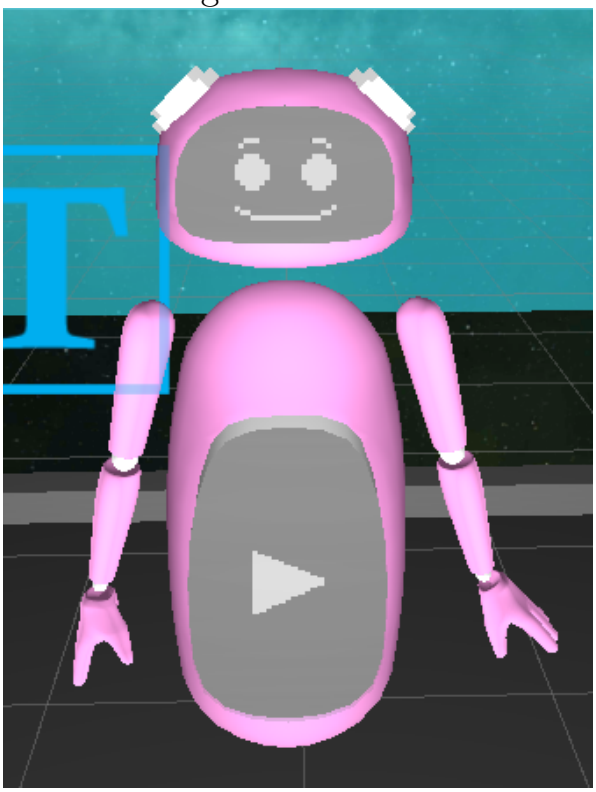
2 Development

This section will provide a description of relevant aspects of the developed application.

2.1 The Environment

The avatar is a robot present in the environment meant to interact with the user to aide their learning of ASL. The avatar has a head with

Figure 1: Avatar

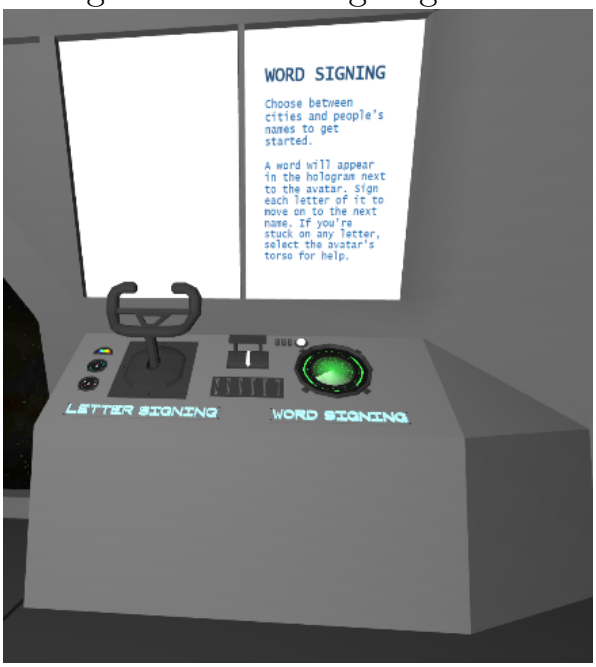


expressions on a facial screen and symbols on a body screen (which is equipped with shapes and buttons to trigger certain game events). The environment is constructed as the interior of a spaceship with the avatar positioned at the front in front of a bay of windows. The user selects game through "control panels" on either side of the space. The new space allows for creativity and buttons are incorporated into the panels for an interactive and immersive game experience. The games are adapted to git the scenery and the environment takes advantage of the VR setting.

We decided to create a friendly-looking pastel-colored robot avatar designed by the DALI team with direction from their partners in order to ensure users of all ages would feel comfortable interacting with it and that it would be easy on the eyes for extended periods of time. We settled on a simple rounded design with no arms or legs with a detatched head. Eventually as we wanted to sign more complex signs, we added arms, hands and fingers, which are all supported by humanoid rigs in Unity.

However, the application currently cannot support facial expressions for ASL, which is a core aspect of understanding the language. This is an area of growth for future designers and developers. The avatar rig's arm, hand and finger joints have been appropriately configured and mapped to the model provided in Unity to allow for easy access to manipulate and read positions and rotations of those joints. This will allow us to record data beneficial in the future.

Figure 2: Word Signing Panels



2.2 Word Signing Game Mode

In order to enter the word signing game, users pan to the right on the Oculus and click the second button on the control panel. When your finger hovers over the button, information about the game mode pops up on the white screen above the panel. To click, users pinch their thumb and pointer finger, prompting 3 more panels to show up in front of the avatar (so the user pans back left). Currently there are multiple different game modes that involve signing words such as fingerspelling, but the game mode involving movement starts when the user clicks on the middle panel. Immediately, the panels disappear, a scoreboard is displayed to the left of the game, and a word is written on the blackboard above the avatar. The user is given 4 seconds (subject to change) to sign the given word. After the allotted time is over, the avatar replays the word back to the user, and then signs the correct version of the word if the user displayed an incorrect sign.

Figure 3: Different Game Modes

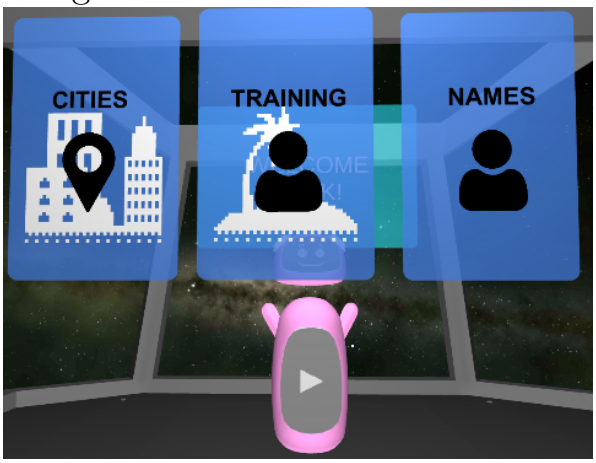
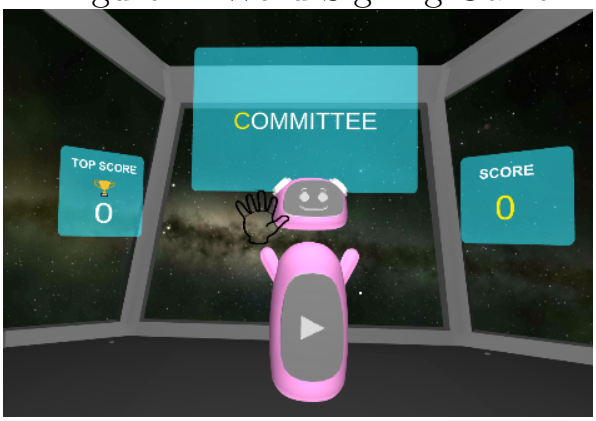


Figure 4: Word Signing Game



Unfortunately, I didn't have enough time to set up the game mode such that the avatar correctly both plays back the user animation as well as shows the correct animation. The code structure is there, but animating the avatar using the correct coordinate space and quaternions was quite difficult, so I left that for future work in order to finish this thesis. Ideally, an implementation would perfectly mimic the avatar so the user can visualize what their motion looks like, but since obviously different people have different hand shapes from each other and from the avatar itself, some calculations will likely be required to create smooth transitions and animations.

2.3 Models and External Software

2.3.1 Word Model

I've created a class to represent a word in ASL with a few intrinsic properties mapping to characteristics of a sign. rotationData is a list of dictionaries mapping either a user's left or right hand to a dictionary of finger bones to floats representing quaternions. positionData is similarly constructed but points to a float representing vectors indicating the bone's position in the coordinate space relative to the bone's parent. The position data of user fingers is unused in this project. Then we have wristPositionData which is a list of dictionaries mapping a user's left or right hand to a vector float representing the user's wrist position in the world coordinate space relative to the face of a user. wristRotationData is similarly constructed but points to a float representing a quaternion in word space. Each one of these data structures is firstly contained within a list. This list represents the position or rotation of the relevant variable at a snapshot in time. The length of these data structures is the number of frames captured by the application. frameSplitBreakdowns is a list of floats representing the second that each of the frames was taken. This is used in the future to normalize time signatures between different word trajectories.

2.3.2 Trajectory Model

A Trajectory is a representation of a Word with reformatted data used for analysis. leftHandFingerBoneRotationData and rigthHandFingerBoneRotationData are created through Word.rotationData and are each a list of dictionaries mapping finger bone to quaternion. leftHandWristRotationData and rightHandWristRotationData are both taken from Word.wristRotationData and are both lists of quaternions representing the rotation of each wrist in the world coordinate space. leftHandLocationWithRespectToFace and rightHandLocationWithRespectToFace are created from wristPositionData are both lists of vectors representing floats representing the position of each wrist in world space relative to the user's face. rightHandUnitVelocityVector and leftHandUnitVelocityVector are a list of floats representing the change in direction and magnitude of motion from each frame to the next. This variable is cre-

ated from the Trajectory.leftHandLocationWithRespectToFace and Trajectory.rightHandLocationWithRespectToFace variables. rightHandPositionWithRespectToleftHand is a list of vectors representing the right hand's position in relation to the left hand and is created from subtracting each hand's position from each other at the same frame. rigtHandWRTLeftHandUnityVelocityVector is a list of vectors representing the direction of motion at each frame for the right hand's position with respect to the left hand. framePointsForComparison is a list of numbers representing a frame within the allotted 4 seconds. This variable is used for trajectory normalization which will be discussed further in detail later in the paper. frameSplitsBreakdown is identical to Word.frameSplitsBreakdown.

2.3.3 Database Manager

The application is connected to a MongoDB instance with various collections. After the time period for the user to sign is up, the application collects the relevant data, transforms it in to the necessary coordinate spaces, then uploads it to the database. The script that determines the similarity between trajectories is written in python and separate from the application. Eventually, this logic should be ported over to the application to allow real time trajectory comparison, but I left that up for future work.

3 Similarity Comparison

This section is devoted towards explaining the implementation of the trajectory similarity comparison technique used in this project.

3.1 Existing Techniques

As you may be able to imagine, trajectory similarity is a problem that has been studied for decades. As the technology for tracking moving objects becomes faster, cheaper and more accurate, we have seen increased volumes of well-obtained data mapping moving objects [13].

Additionally, tracking the similarities of trajectories has been studied to detect collision detection, analyze maritime traffic management, variation of temperature over time and more. Recently, these techniques have been applied to classifying and comparing hand signs, whether in ASL or other visual languages.

3.1.1 Fréchet Metric

The Fréchet metric (or distance) is one of the most popular similarity trajectory measures [14]. The metric was first created by M. Fréchet [15] and can be applied to either continuous or discrete curves. The classic example provided depicts a person walking a dog on the leash. Both the person and the dog are able to vary their speeds, stop, but cannot go backwards. The metric is the minimum leash length required for the person and the dog to complete their trajectories. The algorithm used to determine the values of the distances can be adopted for each specific use case, but a common one used is Euclidean distance. The points are not matched together, meaning it will correctly work for varying sampling rates and trajectory lengths. The worst case time complexity of this metric is $O((m^2k + k^2)logmk)$. Where m and k are the lengths of the trajectories [17].

3.1.2 Dynamic Time Warping (DTW)

Dynamic Time Warping (DTW) is a similarity measure the matches points within the two trajectories. The trajectories are "warped" in a non-linear way to measure similarity while varying sample rates [16].

For trajectories T_A and T_B with lengths m and k, a m X k matrix can be created where each point (i, j) represents the distance between points T_{A_i} and T_{B_j} [18]. A warping path W is created by starting at point (1,1) and incrementing i or j by 1 until you reach (m, k). Therefore we can conceive of the warping path W as a sequence of grid points. Exponentially many path can be created to satisfy that constraint [19]. Then we calculate distances between each of those points in the trajectory using whatever algorithm appropriate for the use case [18]. In this paper, we use euclidean distances between vectors and quaternions. These costs

are summed to get a DTW distance for warping path W. The DTW similarity value is the minimum of all possible DTW distances for all possible warping paths [19]. We can therefore map a single point in a trajectory to multiple points on the other trajectory, allowing us to deal with different lengths and varying sampling rates. The time complexity of finding the DTW cost for a path W is $O(mk)$ [18].

3.1.3 Longest Common Subsequence (LCSS)

Longest Common Subsequence is a similarity technique where some points are able to remain unmatched. The output value represents a count of the maximum number of points between trajectories that can be considered identical. The trajectories are traversed from start to end [20]. The measure generally functions well with different sampling rates; however, widely varying rates may cause issues when many points are left unmatched. Using dynamic programming, the time complexity of this algorithm is $O((m + k)\delta$ where δ is a constant [21].

3.1.4 Edit Distance

At its core, this algorithm counts the minimum number of edits required to make two trajectories identical. The method doesn't require equivalent length trajectories, but this will inflate the actual edit distance since points would either need to be removed or added to match lengths. The time complexity of this algorithm is $O(mk)$ [22].

3.1.5 Our Use Case

Previous literature has been published where academics classify and recognize hand gestures using Dynamic Time Warping (DTW) [23, 24, 25, 26]. We use these papers as a guide towards evaluating the trajectories created by the Oculus Quest. DTW requires no training, making it perfect for our small sample set. However, this is an area for future work. With more time, we could conduct a comparison of trajectory similarity techniques and use the best one for the application.

3.2 Data Manipulation

3.2.1 Data Gathering

C-sharp and Unity provides us with many useful APIs for retrieving and manipulating bone data. We can find finger bone rotations and positions and hand rotations and positions which are essential towards modeling ASL. Additionally, we can retrieve the time the data was found, giving us the ability to calculate velocity, allowing us to account for speed. I record all relevant data through the application, upload it to a database, then retrieve it using a python script and finally normalize the data.

3.2.2 Definitions

The following are attributes of a Trajectory represented by the model described above where i represents the ith frame for each frame:

1. leftHandFingerBoneRotationData := $LF_iR(i)$
2. leftHandUnitVelocityVector := $LUV(i)$
3. rightHandFingerBoneRotationData := $RF_iR(i)$
4. rightHandUnitVelocityVector := $RUV(i)$
5. leftHandWristRotationData := $LWR(i)$
6. rightHandWristRotationData := $RWR(i)$
7. leftHandLocationWithRespectToFace := $LF_aP(i)$
8. rightHandLocationWithRespectToFace := $RF_aP(i)$
9. rightHandPositionWithRespectToLeftHand := $RLP(i)$
10. rightHandWRTLeftHandUnitVelocityVector := $RLV(i)$
11. framePointsForComparison := $a[i]$
12. frameSplitsBreakdown := $b[i]$

Due to normalization, the length of all these variables besides b will be length n. As discussed in the next section, we use b to interpolate new lists for each variable. We can establish a feature set $F := \{LF_iR(i), LUV(i), RF_iR(i), ..., RF_aP(i), RLP(i), RLV(i)\}$ that defines and characterize a trajectory T.

3.2.3 Time Normalization

framePointsForComparison, as mentioned earlier is an array a with varying length n with $a[i]$ - $a[i-1]$ is equal to $4/n$ because 4 is the amount of seconds given to a user to sign. This array is used to normalize all the trajectories since they are bound to be less than 4 seconds. For each Trajectory T that has n frames over 4 seconds, I create a new quaternion or vector for each of the variables in the trajectory model that represents that variable's value at frame/second $a[i]$. For example, if n was 50, $|a|$ = 50 and it looks like the following: $[0.08, 0.16, 0.24, ...]$.

I use frameSplitsBreakdown (array represented as b) to find out for each variable in each trajectory what index j is where $b[j] < a[i]$. We can use linear interpolation between $b[j]$ and $b[j+1]$ to find percentage completed p with the formula

$$p_i = \frac{a[i] - b[j]}{b[j+1] - b[j]}$$

Then, we create a new vector or quaternion using p_i. For example, assuming we have vectors v_j and v_{j+1}, we create a new vector v_i:

$$v_i = (v_j.x + p_i*(v_{j+1}.x - v_j.x), v_j.y + p_i*(v_{j+1}.y - v_j.y), v_j.z + p_i*(v_{j+1}.z - v_j.z))$$

We do the same construction for quaternions, which is the same with the inclusion of a w dimension. After calculations, I add the created variable to a temporary version of each trajectory feature which will eventually replace the feature itself after normalization is complete. After this algorithm, all variables will be lists of length n, providing us an easy way to compare trajectories.

3.2.4 Velocity Attribute Creation

$RLP(i)$ is created by looping through all frames within $LF_aP(i)$ and

$RF_aP(i)$ and getting the difference. Formally, for $i...n$:

$$RLP(i) = LF_aP(i) - RF_aP(i)$$

$LUV(i)$, $RUV(i)$, $RLV(i)$ are each created from $LF_aP(i)$, $RF_aP(i)$, $RLP(i)$ respectively. For simplicity sake, we will represent the velocity vector attributes as UV and the attributes they were created from as A. Following this, we can define UV [24]:

$$UV_i = \frac{A[i+1] - A[i-1]}{||A[i+1] - A[i-1]||}$$

Remember that after normalization all attributes will have length n. I add a vector representing 0 velocity to the beginning and end of each unit velocity attribute to force the size of the list to n. $||A[i+1] - A[i-1]||$ denotes the euclidean magnitude of $A[i+1] - A[i-1]$, which assuming we are evaluating a new vector v, is calculated as follows:

$$e = \sqrt{v.x^2 + v.y^2 + v.z^2}$$

Quaternion euclidean distances are similarly constructed with the inclusion the w dimension. This concludes any data manipulation required to move forward with DTW.

3.3 DTW Algorithm

Assuming we are evaluating the similarity between a trajectory Q and trajectory X, the following is psuedocode for the implementation of DTW:

Algorithm 1 Calculate DTW cost

Require: $|Q| = |X|$

$c \leftarrow 0$

for i in X or $Q.a$ **do**

for A in F **do**

$c = c + \text{euclideanDistance}(Q.LF_aP(i) - X.LF_aP(i))$
$+ \text{euclideanDistance}(Q.LF_iR(i) - X.LF_iRP(i))$
$+ \text{euclideanDistance}(Q.LUV(i) - X.LUV(i))$
$+ \text{euclideanDistance}(Q.RF_iR(i) - Q.RF_iR(i))$
$+ \text{euclideanDistance}(Q.RUV(i) - X.RUV(i))$
$+ \text{euclideanDistance}(Q.LWR(i) - X.LWR(i))$
$+ \text{euclideanDistance}(Q.RWR(i) - X.RWR(i))$
$+ \text{euclideanDistance}(Q.RF_aP(i) - X.RF_aP(i))$
$+ \text{euclideanDistance}(Q.RLP(i) - X.RLP(i)) +$
$\text{euclideanDistance}(Q.RLV(i) - X.RLV(i))$

end for

end for

return c

Euclidean distance calculations are the same as described in earlier sections.

4 Testing

This section provides a description of how I tested DTW and an analysis of its effectiveness.

4.1 Sign Gathering

I chose 5 pairs of signs that are similarly signed [27] and signed each of them twice for a 20 independent trajectories. I chose not to discriminate between 1-handed and 2-handed signs. The following is a list of the pairs of signs used:

1. ASK and QUESTION
2. ATTENTION and FOCUS

3. GLASSES and GALLAUDET
4. SAD and FRIENDLY
5. SENATE and COMMITTEE

4.2 Conceptual Framework

In order to evaluate the effectiveness of DTW, I ran trajectory comparisons between each sign including the pair of signs with duplicate words. I didn't run trajectories against themselves because they are identical, and the algorithm returns a cost of 0.

I believe this framework of comparison allows us to examine multiple cases of interest in evaluating the DTW cost:

1. When signing duplicate words with each other, we expect the lowest cost c_{dup} of signing that word with any other word.
2. When signing words that are similarly signed (as defined by the pairs), we would expect a cost $c_{similar} > c_{dup}$.
3. When signing words that are not similarly signed and that are not duplicates, we expect a cost $c_{unrelated} > c_{similar}$

Therefore we create an expectation of $c_{unrelated} > c_{similar} > c_{dup}$ where c is the DTW cost between words.

4.3 Results

At varying time normalization levels, $n = \{10, 25, 50, 100\}$ we present DTW distances as describes in the above subsections. The following figures describe my results:

c_{dup}, $c_{similar}$ and $c_{unrelated}$ are represented by orange, yellow and light blue cells respectively. Additionally, these numbers are averages of all possible trajectory comparisons for the given words in the table.

Figure 5: DTW Distances at $n = 10$

	ASK	QUESTION	ATTENTION	FOCUS	GLASSES	GALLAUDET	SAD	FRIENDLY	SENATE	COMMITTEE
ASK	319.53	196.85	281.03	290.75	265.29	226.43	286.53	290.97	222.19	262.93
QUESTION		174.77	341.02	349.27	326.21	254.39	340.26	341.95	236.16	283.92
ATTENTION			60.02	93.83	109.0	186.0	89.44	126.19	243.31	161.67
FOCUS				77.27	108.37	180.29	94.17	96.22	243.83	167.84
GLASSES					68.64	141.7	110.81	107.18	208.88	141.01
GALLAUDET						115.88	182.85	166.21	141.53	159.05
SAD							56.5	109.55	249.25	161.66
FRIENDLY								41.77	226.34	163.51
SENATE									121.68	185.0
COMMITTEE										127.97

Figure 6: DTW Distances at $n = 25$

	ASK	QUESTION	ATTENTION	FOCUS	GLASSES	GALLAUDET	SAD	FRIENDLY	SENATE	COMMITTEE
ASK	806.45	505.48	721.41	748.69	679.6	576.63	738.18	743.71	568.9	672.07
QUESTION		441.27	873.97	895.48	830.77	649.1	876.43	873.83	607.26	727.47
ATTENTION			147.55	240.35	281.25	481.25	234.33	326.05	627.74	414.56
FOCUS				206.38	284.75	469.97	247.16	254.33	626.12	431.09
GLASSES					184.42	372.64	295.22	284.52	534.62	357.36
GALLAUDET						302.72	482.67	430.49	369.95	416.8
SAD							167.52	291.87	642.27	424.38
FRIENDLY								118.35	582.08	413.92
SENATE									323.16	466.49
COMMITTEE										333.27

Figure 7: DTW Distances at $n = 50$

	ASK	QUESTION	ATTENTION	FOCUS	GLASSES	GALLAUDET	SAD	FRIENDLY	SENATE	COMMITTEE
ASK	1613.53	1016.31	1455.61	1512.59	1375.81	1165.31	1489.11	1501.81	1144.55	1356.7
QUESTION		888.35	1753.44	1804.23	1675.87	1309.18	1767.06	1759.75	1216.95	1462.65
ATTENTION			328.8	506.11	577.55	976.95	487.88	670.12	1267.18	835.53
FOCUS				424.22	578.9	947.75	510.26	526.34	1267.49	871.31
GLASSES					377.71	758.69	603.74	584.51	1083.42	722.93
GALLAUDET						615.22	978.04	869.22	748.33	846.7
SAD							349.2	599.87	1302.33	858.79
FRIENDLY								252.44	1181.83	838.31
SENATE									652.98	949.36
COMMITTEE										681.35

4.3.1 Time Normalization Analysis

Looking at the figures, we can identify patterns within our results and investigate whether they conform to our expectations. However, we can conclude that since the patterns are functionally identical between different levels of n, the time normalization does not affect the relationships within the data. As long as developers use consistent levels of n, we can compare costs between each other.

4.3.2 ASK_1 vs ASK_2

For each time normalization length for the word ask, the $c_{dup} > c_{similar} > c_{unrelated}$ is always true. There are a number of possible explanations. Simply, assuming our methodology is correct, that would indicate that one or either both of the hand signs for ASK were signed incorrectly. When inspecting the data closer (at $n = 50$) we see four different scores reported for the ASK vs. ATTENTION trajectory comparison: 840.2, 858.16, 2031.58, 2092.52. Due to the placement in the output and the structure of the word array we loop through, we can separate the costs into two clusters. As we can see, one of the clusters has costs describing the similarity of ASK_1 to QUESTION to be much closer than ASK_2 to QUESTION. The following is a table of DTW distances excluding ASK signs:

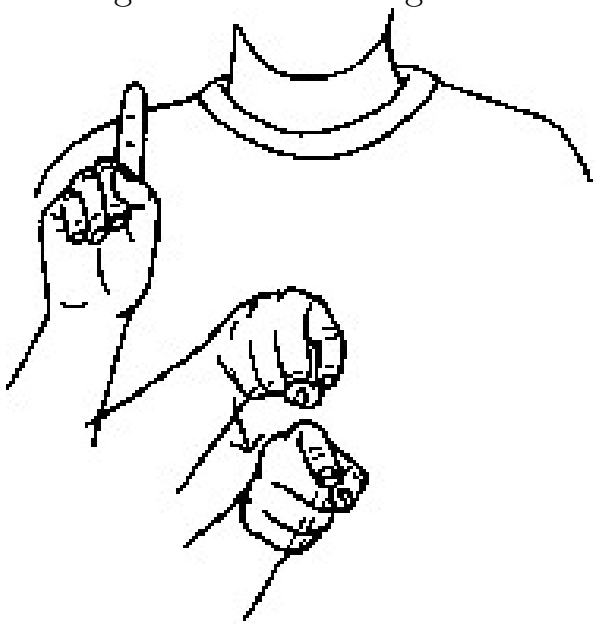
Figure 8: n=50, without ASK

	QUESTION	ATTENTION	FOCUS	GLASSES	GALLAUDET	SAD	FRIENDLY	SENATE	COMMITTEE
QUESTION	888.35	1753.44	1804.23	1675.87	1309.18	1767.06	1759.75	1216.95	1462.65
ATTENTION		328.8	506.11	577.55	976.95	487.88	670.12	1267.18	835.53
FOCUS			424.22	578.9	947.75	510.26	526.34	1267.49	871.31
GLASSES				377.71	758.69	603.74	584.51	1083.42	722.93
GALLAUDET					615.22	978.04	869.22	748.33	846.7
SAD						349.2	599.87	1302.33	858.79
FRIENDLY							252.44	1181.83	838.31
SENATE								652.98	949.36
COMMITTEE									681.35

The reported accuracy rate for this test is 91.67%, which is significantly better and likely evidence of my incorrect signing of the first ASK word.

If we inspect $c_{similar}$ values across time normalization levels, we find an identical pattern. Therefore, we can reason that one of the signs for ASK was either signed incorrectly or the data was not collected accurately. Another explanation is based on occlusion regarding ASK. The following image is a representation of how to sign ASK:

Figure 9: How to Sign ASK



From examining the image, you can see from the perspective of the face, the end finger joint positions and rotations for the pointer finger are occluded by the hand since they are in between the joints and the face. This can lead to improper measurement and inaccurate vectors or quaternions in world space, artificially inflating DTW cost.

4.3.3 c_{dup}

Besides ASK, $c_{dup} > c_{similar}$ and $c_{dup} > c_{unrelated}$ remain true. Though $c_{similar} > c_{unrelated}$ is not always true, it is true in most circumstances. Regardless, we are mostly interested in c_{dup} being the minimum DTW cost for all trajectory comparisons.

4.3.4 Accuracy Rates

We define an error rate to be the number of times the expectations are satisfied divided by the total number of comparisons to process. To clarify, we expect orange cells to be the lowest value for their row and column, and yellow cells to be the lowest value excluding orange cells at their row and column. At all time normalization levels, the accuracy rate

is 83.85%, and the same cells are incorrect.

5 Conclusion

This section concludes the paper, describes the implications of our findings, and guides direction for future work on this application.

5.1 Implications

Even if our analysis of the c_{dup} value for ASK_1 vs. ASK_2 is incorrect, the accuracy rate of our implementation is 84%. Alternatively, if I simply incorrectly had signed one of the ASK words, then we can expect the accuracy rate to improve. Furthermore, we can glean the pedagogical value of the application if in even a small dataset we are able to determine that one of the words was signed incorrectly. Additionally, the reported accuracy rate includes comparisons between $c_{similar}$ and $c_{unrelated}$ which is not as important as the expectation the c_{dup} is the minimum value. Given that we fulfill that expectation 100% of the time excluding ASK_1 vs. ASK_2, we've successfully been able to validate signs using DTW.

5.2 Future Work

The following is a list of future work that would move this paper and application along:

1. Finish animating the avatar to mimic user signs and show validated signs after user prompt ends.

2. Implement and compare the effectiveness of different trajectory similarity measures.

3. Delineate signs by them being 1-handed or 2-handed and compare signs within and across those classes to identify whether that makes a difference. Other papers have implemented DTW using this methodology [24].

4. Implement machine learning techniques to evaluate signed sentences.

5. Deal with the occlusion of specific joints.

References

[1] Kemp, Mike. "Why is Learning American Sign Language a Challenge?" American Annals of the Deaf. July 1998. Vol. 143, No. 3.

[2] "Mind the Gap!" Communication and the Educational Relation Author(s): Gert Biesta Source: Counterpoints , 2004, Vol. 259, No Education Without Relation (2004), pp. 11-22 Published by: Peter Lang AG Stable URL: https://www.jstor.org/stable/42978490

[3] Byram, Katra, and Claire Kramsch. "Why is it so difficult to teach language as culture?." The German Quarterly 81.1 (2008): 20-34.

[4] McKee, Rachel Locker, and David McKee. "What's so hard about learning ASL?: students' teachers' perceptions." Sign Language Studies 75.1 (1992): 129-157.

[5] Mills, Carol Bergfeld. "Factors influencing manual sign learning in hearing adults." Sign Language Studies (1984): 261-278.

[6] Schornstein, Ruth Ann. "Teaching ASL in the university: One teacher's journey." Sign Language Studies 5.4 (2005): 398-414.

[7] Bahrani, Taher, and Tam Shu Sim. "Informal Language Learning Setting: Technology or Social Interaction?." Turkish Online Journal of Educational Technology-TOJET 11.2 (2012): 142-149.

[8] Adams, EDWARD N., H. W. Morrison, and J. M. Reddy. "Conversation with a computer as a technique of language instruction." The Modern Language Journal 52.1 (1968): 3-16.

[9] Hubbard, Philip L. "Language Teaching Approaches, the Evaluation of CALL Software, and Design Implications." (1982).

[10] Salaberry, M. Rafael. "The use of technology for second language learning and teaching: A retrospective." The modern language journal 85.1 (2001): 39-56.

[11] Snoddon, Kristin. "Technology as a learning tool for ASL literacy." Sign Language Studies 10.2 (2010): 197-213.

[12] Brega, Jose Remo Ferreira, et al. "A virtual reality environment to support chat rooms for hearing impaired and to teach Brazilian Sign Language (LIBRAS)." 2014 IEEE/ACS 11th International Conference on Computer Systems and Applications (AICCSA). Ieee, 2014.

[13] Toohey, Kevin, and Matt Duckham. "Trajectory similarity measures." Sigspatial Special 7.1 (2015): 43-50.

[14] J. Gudmundsson, P. Laube, and T. Wolle. Computational movement analysis. In Springer handbook of geographic information, pages 423–438. Springer, 2012.

[15] M. M. Fréchet. Sur quelques points du calcul fonctionnel. Rendiconti del Circolo Matematico di Palermo, 22(1):1–72, 1906.

[16] Yuan, Yu. Image-based gesture recognition with support vector machines. University of Delaware, 2008.

[17] H. Alt and M. Godau. Computing the Fréchet distance between two polygonal curves. International Journal of Computational Geometry and Applications, 5(01n02):75–91, 1995.

[18] D. Berndt and J. Clifford. Using dynamic time warping to find patterns in time series. In KDD workshop, volume 10, pages 359–370. Seattle, WA, 1994.

[19] E. Keogh and C. A. Ratanamahatana. Exact indexing of dynamic time warping. Knowledge and Informa- tion Systems, 7(3):358–386, 2005.

[20] M. Vlachos, G. Kollios, and D. Gunopulos. Discovering similar multidimensional trajectories. In Proc. 18th International Conference on Data Engineering (ICDE), pages 673–684, 2002.

[21] P. Ranacher and K. Tzavella. How to compare movement? A review of physical movement similarity mea- sures in geographic information science and beyond. Cartography and Geographic Information Science, 41(3):286–307, 2014.

[22] L. Chen, M. T. Özsu, and V. Oria. Robust and fast similarity search for moving object trajectories. In Proc. ACM SIGMOD International Conference on Management of Data, SIGMOD '05, pages 491–502, New York, NY, USA, 2005. ACM.

[23] Wang, Haijing, et al. "A system for large vocabulary sign search." European Conference on Computer Vision. Springer, Berlin, Heidelberg, 2010.

[24] Jangyodsuk, Pat, Christopher Conly, and Vassilis Athitsos. "Sign language recognition using dynamic time warping and hand shape distance based on histogram of oriented gradient features." Proceedings of the 7th International Conference on PErvasive Technologies Related to Assistive Environments. 2014.

[25] A. Corradini. Dynamic time warping for off-line recognition of a small gesture vocabulary. In Recognition, Analysis, and Tracking of Faces and Gestures in Real-Time Systems, 2001. Proceedings. IEEE ICCV Workshop on, pages 82–89. IEEE, 2001.

[26] G. Ten Holt, M. Reinders, and E. Hendriks. Multi-dimensional dynamic time warping for gesture recognition. In Thirteenth annual conference of the Advanced School for Computing and Imaging, volume 300, 2007.

[27] Shaw, E. Delaporte, Y. (2014). A Historical and Etymological Dictionary of American Sign Language. Washington: Gallaudet University Press.