



UNIVERSITÀ DEGLI STUDI DI CATANIA
DIPARTIMENTO DI MATEMATICA E INFORMATICA
CORSO DI LAUREA TRIENNALE IN INFORMATICA

Elenoire Scaletta

Missing Keypoints Reconstruction in Human Pose
Estimation via Interpolation Algorithms

REPORT ON DIGITAL FORENSICS PROJECT

Professors:
Sebastiano Battiato,
Luca Guarnera

Academic Year 2023 - 2024

Contents

1	Introduction	1
2	Prerequisites	2
2.1	Interpolation Methods	2
2.2	Distribution Evaluation Metrics	5
2.3	Models	6
2.4	Python and Related Libraries	7
3	Dataset	9
4	Method	12
4.1	Preprocess Dataset Operations	12
4.1.1	Ground Truth Dataset Creation	12
4.1.2	Zero Sequences Generation	13
4.2	Interpolations Applied	14
5	Experiments and Outcomes	15
5.1	Results of Interpolations Performance on Reference Data . . .	15
5.1.1	Reconstructed Joints Visual Inspections	16
5.1.2	Methods Distributions Trends Analysis	18
6	Conclusions	21

Chapter 1

Introduction

In this work, we focus on Human Body Pose Estimation (HBPE), a field dedicated to determining the pose of a human body by estimating the 2D or 3D spatial positions of its joints. HBPE is crucial in various applications, including motion capture, activity recognition, and human-computer interaction.

The detection of body joints in a video or a single frame is influenced by various factors such as lighting conditions, occlusions, and undetected joint positions. These challenges often result in incomplete or inaccurate pose estimations. To address these issues, we aim to estimate the positions of missing joints using different interpolation algorithms.

This work provides a comprehensive overview of estimating missing joint spatial information through four distinct interpolation methods. Each interpolation method will be thoroughly examined and employed to estimate the missing values, offering a detailed analysis of their effectiveness and accuracy.

The process begins with the selection of a dataset from Kaggle, further discussed in Chapter 3, which represents various human actions. The dataset contains human body keypoints extracted using OpenPose on a frame-by-frame basis. To ensure a reliable ground truth for quantitative analysis, we retrieved a subset of the dataset by removing entries with any missing values as treated in Chapter 4. From this refined dataset, we intentionally set certain values to zero based on specific criteria. These zeroed values were then estimated using different interpolation methods. Various qualitative and quantitative inspections followed in Chapter 5 to determine the most precise interpolation method for approximating the ground truth values.

Through this research, we seek to enhance the reliability and accuracy of pose estimation systems, contributing to the advancement of technologies that rely on precise human body tracking.

Chapter 2

Prerequisites

In this section we present the interpolation algorithms used, the evaluation metrics, the OpenPose model utilized to create the dataset and the main libraries involved in the work.

2.1 Interpolation Methods

In this project, our main focus is on interpolation, it is like a way to connect the dots when we have some data points, but there are gaps in between. We are using this technique to estimate what is happening with an human's body joints when we do not have all the data. Sometimes, the pose estimator that we use to detect these joints does not work perfectly, leaving us with missing data at certain points in time or positions. So, interpolation comes to the rescue. It helps us smooth out these gaps and create a complete picture of what is happening over time.

The interpolations used in this work are the following:

- Nearest Neighbor
- Linear
- Inverse Distance Weight
- Spline

Nearest Neighbor

Nearest Neighbor (Nearest) interpolation is one of the simplest interpolation techniques. It works by assigning the value of the nearest data point to the location we want to interpolate.

Formally, given a set of data points (x_i, y_i) , where $i = 1, 2, \dots, n$ represents the known data points, and we want to estimate the value y at a specific location x , Nearest Neighbor Interpolation can be defined as:

$$\text{Nearest Neighbor}(x) = y_k, \text{ where } k = \arg \min_i |x - x_i| \quad (2.1)$$

where:

x : The location where we want to estimate the value.

x_i : The x -coordinates of the known data points.

y_i : The corresponding y -values.

k : The index of the nearest data point to x , found by minimizing the absolute difference between x and x_i .

y_k : The value at the nearest data point, which is used as the interpolated value at x .

In practice, this will result in a step function.

Linear

Linear interpolation is a straightforward method to estimate values between two known data points. It assumes that the relationship between data points is approximately linear. In other words, it draws a straight line between two adjacent data points and calculates the value at a desired position along that line.

Suppose we have two data points, (x_0, y_0) and (x_1, y_1) , and we want to estimate a value y at a position x that falls between x_0 and x_1 . Linear interpolation can be expressed as:

$$y = y_0 + \frac{(x - x_0)}{(x_1 - x_0)} \cdot (y_1 - y_0) \quad (2.2)$$

Here is what each variable represents:

- y : The estimated value at the position x within the range $[x_0, x_1]$.
- x_0 : The x-coordinate of the first data point.
- y_0 : The y-coordinate of the first data point.
- x_1 : The x-coordinate of the second data point.
- y_1 : The y-coordinate of the second data point.
- x : The position where we want to estimate the value y .

Inverse Distance Weight

Inverse Distance Weight (IDW) interpolation is a method used to estimate values at unmeasured locations based on values at nearby measured locations.

It assumes that values at a given location are influenced by the values at other locations, with the influence decreasing as the distance between the locations increases.

In mathematical terms, the formula for IDW can be expressed as follows:

The interpolated value at a location (x, y) can be denoted as z_0 , and it is calculated as a weighted average of the known values at nearby locations. The weight assigned to each known value is inversely proportional to the distance between the unknown location and the known location. The general formula for IDW interpolation is:

$$z_0 = \frac{\sum_{i=1}^n \frac{z_i}{d_i^p}}{\sum_{i=1}^n \frac{1}{d_i^p}} \quad (2.3)$$

where:

z_0 is the interpolated value at the location (x, y) .

z_i is the known value at the i th location.

d_i is the distance between the interpolated location (x, y) and the i th known location.

p is a user-defined positive power parameter that controls the influence of distance. Common values for p are 1 (inverse linear distance) and 2 (inverse squared distance).

n is the total number of known locations used in the interpolation.

This formula essentially calculates a weighted average of the known values, with the weights based on the inverse of distances raised to power p . The larger the distance, the smaller the weight, and vice versa. The power parameter p controls the rate at which the influence of distance diminishes.

Spline

In Spline interpolation, we aim to approximate a function or a curve, $f(x)$, using piecewise defined cubic polynomials. These cubic polynomials, known as “spline functions” are combined to create a smooth and continuous curve. The spline passes through given data points (x_i, y_i) for $i = 0, 1, 2, \dots, n$.

A cubic spline consists of n cubic polynomials, each defined on a subinterval $[x_i, x_{i+1}]$, where i ranges from 0 to $n - 1$. The spline is represented as:

$$S(x) = \begin{cases} S_0(x) & \text{if } x_0 \leq x \leq x_1 \\ S_1(x) & \text{if } x_1 \leq x \leq x_2 \\ \vdots & \\ S_{n-1}(x) & \text{if } x_{n-1} \leq x \leq x_n \end{cases}$$

Each $S_i(x)$ is a cubic polynomial, typically expressed in the form:

$$S_i(x) = a_i + b_i(x - x_i) + c_i(x - x_i)^2 + d_i(x - x_i)^3$$

The coefficients a_i , b_i , c_i , and d_i are determined by imposing various conditions to ensure continuity and smoothness at the data points.

2.2 Distribution Evaluation Metrics

In this section, we provide an overview about the evaluation metrics used to compare the ground truth joints distribution with the interpolated versions.

Euclidean Distance

The Euclidean distance is a fundamental distance metric used to measure the straight-line distance between two points in Euclidean space. In mathematical notation, the Euclidean distance between two points, say A and B , with coordinates (x_1, y_1, z_1) and (x_2, y_2, z_2) , respectively, in three-dimensional space is calculated as:

$$d(A, B) = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2 + (z_1 - z_2)^2}$$

In this formula: $d(A, B)$ represents the Euclidean distance between points A and B . x_1 , y_1 , and z_1 are the coordinates of point A . x_2 , y_2 , and z_2 are the coordinates of point B .

The formula essentially computes the square root of the sum of squared differences in coordinates for each dimension (in this case, x , y , and z). This yields the straight-line distance between the two points in a three-dimensional space.

Root Mean Square Error

The Root Mean Square Error (RMSE) is one of the most commonly used measures for evaluating the quality of predictions. It provides a comprehensive assessment of the accuracy of predictions by quantifying the average deviation between predicted values and actual values in a dataset.

$$\text{RMSE} = \frac{\sqrt{\sum_{i=1}^n (y_i - \hat{y}_i)^2}}{n}$$

Where:

- n is the number of data points.

- y_i is the actual value of the dependent variable for the i^{th} observation.
- \hat{y}_i is the predicted value of the dependent variable for the i^{th} observation.
- The term $(y_i - \hat{y}_i)^2$ represents the squared difference between the actual and predicted values for each observation.

Bhattacharyya Distance

The Bhattacharyya distance is a measure used to quantify the similarity between two probability distributions. Named after the Indian statistician Anil Kumar Bhattacharyya, it is widely used in various fields such as pattern recognition, image processing, and machine learning for tasks like classification and clustering.

Given two probability distributions P and Q over the same domain X , the Bhattacharyya distance is defined as:

$$D_B(P, Q) = -\ln(BC(P, Q)) \quad (2.4)$$

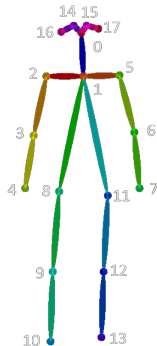
where $BC(P, Q)$ is the Bhattacharyya coefficient, which measures the amount of overlap between the two distributions and is given by:

$$BC(P, Q) = \sum_{x \in X} \sqrt{P(x)Q(x)} \quad (2.5)$$

2.3 Models

OpenPose

OpenPose is an open-source library for multi-person 2D pose detection, including body, foot, hand, and facial keypoints. It is developed by CMU Perceptual Computing Lab. There are several models used in OpenPose, the one employed in this work is COCO.



COCO dataset is a pre-trained deep learning model used in the OpenPose framework for multi-person pose estimation. This model is trained on the COCO (Common Objects in Context) dataset, which contains over 200,000 images with more than 250,000 labeled people. The model is capable of detecting up to 18 key points in each person. These keypoints can be used to estimate

the pose of each person in the image, including their joint angles and body orientation. However, despite its large use, it has not been chosen for this project due to its lower accuracy and fewer keypoints compared to other available models.

2.4 Python and Related Libraries



Python is a high-level, object-oriented, and interpreted programming language (version used: 3.10). Python offers developers many resources to create sophisticated and high-performance applications. Lots of its libraries have been used in this project, including Scipy, PyTorch, Pandas in particular.



SciPy SciPy is an open-source scientific computing library for Python. It builds upon the capabilities of NumPy, another fundamental Python library, by providing additional functionality for a wide range of scientific and engineering applications. Some of the involved modules are: *scipy.interpolate*, *scipy.special*, *scipy.stats*.


The *interpolate* module provides a wide range of interpolation techniques for working with data. These techniques are essential for estimating values between known data points approximating functions from limited data. The module includes functions for linear, nearest, spline and others interpolation.



Pandas is a data manipulation library that offers a variety of powerful tools to quickly and efficiently analyze, clean, and transform data. It provides functionalities such as merging, reshaping, indexing, slicing, and grouping, which are essential for data manipulation. Its primary data structure, DataFrame, is a flexible and robust data structure that allows users to work with data in a tabular form, similar to a spreadsheet. In our project, Pandas has been extensively used to work with CSV files and to manipulate data in some algorithms.



NumPy is a numerical computing library for Python. It provides a multi-dimensional array object, along with a set of mathematical functions to operate on these arrays. It is widely used in scientific and data-intensive computing because of its ability to perform complex mathematical operations efficiently and effectively. It provides a high-performance array computing functionality and tools for working with them, such as linear algebra or random number generation.

 **Matplotlib** Matplotlib is a Python library used for creating static, animated, and interactive visualizations. It offers a range of customization options to make graphs, histograms, scatterplots, and other visualizations.

Chapter 3

Dataset

In this chapter, we discuss about the data we are working on.

The current dataset, is taken from Kaggle at this url <https://www.kaggle.com/datasets/pashupatigupta/human-keypoints-tracking-dataset?resource=download>. There are no significant specifications regarding the dataset, thus some analyses has been manually executed.

The video dataset (not provided) comprises 4 clips showing human actions. Each video has a resolution of 1280x720 at 30 frames per second (fps), as noted in a given sample.

The related frames have been extracted using OpenPose using COCO model. Further explanation are present in 2.3. The total amount of frames contained on it are 2826, with a total amount of 101736 values (last column excluded).

The actions executed by the subjects in the video clips and the related frame occurrences are represented in Table 3.1.

Table 3.1: Occurrences of each action in terms of number of frames.

Actions	Frames
Stand	744
Walk	722
Squat	668
Wave	692
Total	2826

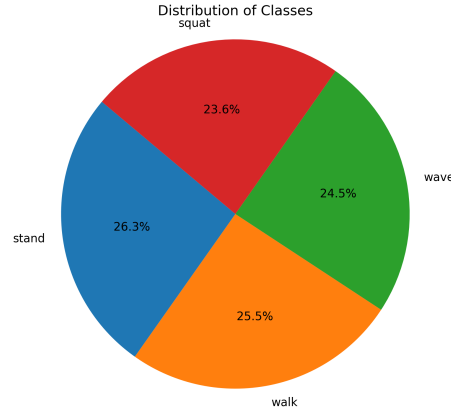


Figure 3.1: Frame classes percentage in the keypoints dataset.

Figure 3.1 shows the piechart of the distribution of frames.

In this dataset, we have already identified some missing values due to camera movements or joint occlusions due to the movement performed.

In order to analyze the number of missing values before processing the dataset, a normalized histogram has been built as illustrated in Fig. 3.2.

This provides insight into the distribution of missing data across various classes, allowing us to understand which activities or movements are more prone to missing frames. Across the considered actions, walking contains the highest number of missing values equal to 317 (representing the 43.91%), due to the joint occlusions of the gait.

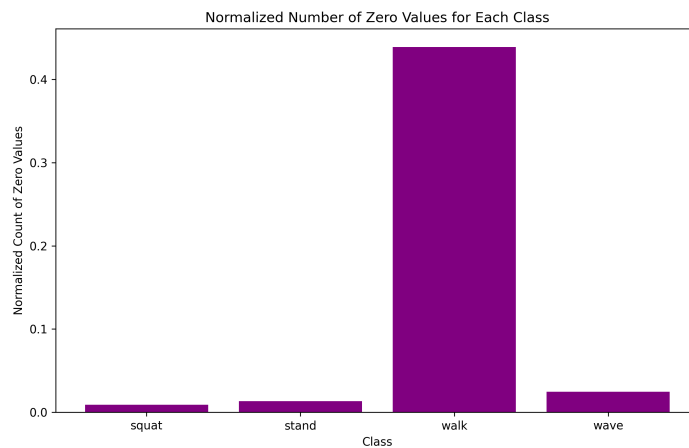


Figure 3.2: Normalized number of zero values in the initial dataset.

The model used is COCO, capable of extracting information about 18 joints from a frame. For each joint, we obtain the X coordinate and Y coordinate. The values for each coordinate have been normalized between 0 (absolute left/up) to 1 (absolute right/down).

The detected joints are as follows:

1. **Nose**
2. **Neck**
3. **Right Shoulder**
4. **Right Elbow**
5. **Right Wrist**
6. **Left Shoulder**
7. **Left Elbow**
8. **Left Wrist**
9. **Right Hip**
10. **Right Knee**
11. **Right Ankle**
12. **Left Hip**
13. **Left Knee**
14. **Left Ankle**
15. **Right Eye**
16. **Left Eye**
17. **Right Ear**
18. **Left Ear**

We have one additional columnn representing the related class at a frame-grain level.

Chapter 4

Method

In this chapter, we explore the methods used to estimate the missing values.

Initially, we preprocess the dataset throughout some operations. Firstly, we added another column which indicates the frame index for each row. Then, we followed some criteria in order to obtain a ground truth subset.

4.1 Preprocess Dataset Operations

In this section, we describe the operations done to obtain a reference dataset, suitable for the following evaluation phase. As aforementioned, we added another column with the related frames, this was needed to make further checks regarding the contiguous sequences considered.

4.1.1 Ground Truth Dataset Creation

We made an assumption about the reference dataset. It cannot contains missing values, because we would miss ground truth values. In order to accomplish this, we both removed those rows with missing values and removed those isolated rows. Isolated rows are those observation which do not link to any previous or subsequent values. They need to be removed because to accurately estimate missing frames, we need them to be contiguous and not sparse. A sequence length parameter is then defined. It represent the minimum length of the sequence, requested to be part of the reference dataset. This has been tuned following a trial and error approach. The trend of Fig. 4.1 shows the correlation between sequence length and number of observations.

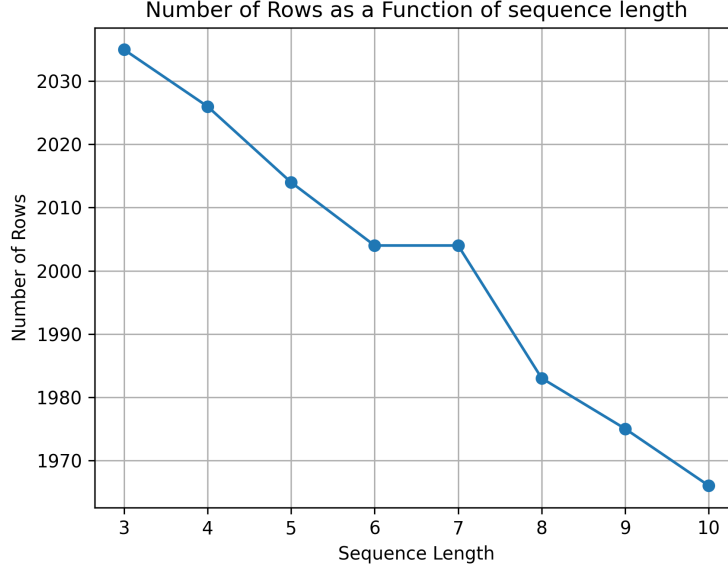


Figure 4.1: Correlation between sequence lengths and number of rows.

The sequence length chosen is equal to 7. In this setup, we have long sequence of value that can be queried. The total number of rows of the updated dataset is equal to 2004, 29.08% smaller than the original.

4.1.2 Zero Sequences Generation

Once the reduced dataset has been obtained, a script has been developed to augment the dataset by inserting zero sequences based on specific criteria.

To simulate missing data, sequences of frames are randomly selected to be set to 0 with a probability of 60%. This random selection is controlled by choosing a random sequence length and a random starting index within valid ranges. However, before setting frames to 0, the code ensures that the selected sequence meets strict validity criteria: it must remain within the same video and maintain temporal adjacency with neighboring frames.

Once a valid sequence is identified, its frames (and corresponding columns, managed by step) are set to 0. This process effectively introduces missing segments into the dataset while attempting to preserve the overall structure and integrity of the video data.

Fig. 4.2 illustrates the occurrences of sequence lengths. The long tail is wanted, considering that the elements of long sequences are harder to estimate in respect of shorter ones.

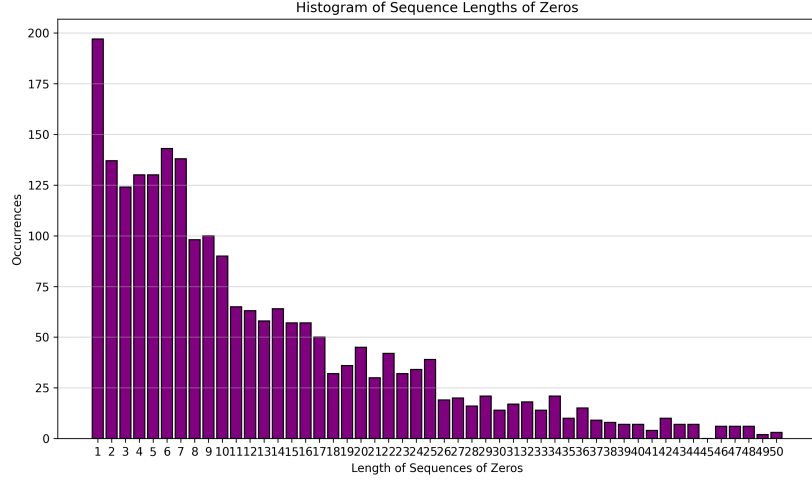


Figure 4.2: Occurrences of sequence lengths.

The total values set to zeros are: 61,304, being the 60.25% of the entire dataset.

The results of the accuracy reached with the various interpolations will be discussed in Chapter 5.

4.2 Interpolations Applied

The obtained dataset, containing a huge portion of zero value sequences, is then used to perform interpolation on it. The interpolation methods that we will use, have been presented and described in Chapter 2. These missing value sequences are vertically aligned within the columns of the dataset. In Chapter 5 we will discuss about the difference among the interpolation methods through quantitative and qualitative analyses.

Chapter 5

Experiments and Outcomes

In this chapter, we look at the analyses and the subsequent outcomes obtained.

We then, consider both quantitative and qualitative analyses.

5.1 Results of Interpolations Performance on Reference Data

The interpolations and the evaluation metrics considered are the ones listed in Chapter 2.

Table 5.1: Interpolation comparisons in relation with ground truth.

Method	Euclidean Distance	RMSE	Bhattacharyya Distance
IDW	10.305	42.2126	0.9224
Linear	6.772	27.739	0.3913
Spline	11.512	47.1567	1.3117
Nearest	8.079	33.096	0.5590

Table 5.1 presents a comparison of interpolation methods in relation to their performance about the ground truth. The table evaluates the effectiveness of different interpolation techniques by measuring their Euclidean Distance, RMSE and Bhattacharyya Distance divergence when compared to the ground truth data.

All values considered have been multiplied by 1000 in order to obtain higher sensitivity, translated in bigger difference between methods.

It is important to note that the Bhattacharyya is a quantity which represents a notion of similarity between two probability distributions.

Among these methods, Linear interpolation stands out with the lowest Euclidean Distance of 6.772, indicating its ability to closely approximate the ground truth data. Additionally, it demonstrates the lowest RMSE value of 27.739 and the smallest Bhattacharyya distance of 0.3913, further underlining, its effectiveness in accurately representing the original data points.

On other hand, Nearest Neighbor has a low Euclidean Distance (8.079) but higher RMSE (33.096) and Bhattacharyya Distance (0.5590), indicating close but less optimal interpolations. IDW shows moderate performance with Euclidean Distance (10.305), RMSE (42.2126), and Bhattacharyya Distance (0.9224), suggesting less accuracy and optimal error distribution.

Of these, Spline interpolation performs worst with the highest Euclidean Distance (11.512), RMSE (47.1567), and Bhattacharyya Distance (1.3117).

These results offer analytical insights into the performance of different interpolation techniques, particularly in approximating joint positions.

However, to further validate the observations made, in the next subsection, we will discuss about some visual inspections regarding both points in 2D space and distribution trends.

5.1.1 Reconstructed Joints Visual Inspections

Now, we examine a frame within the ground truth dataset where all joints are known, allowing for a visual comparison between the ground truth and the reconstructed body joints produced by different interpolation methods. The row considered is the 739 of the video ‘walk’.

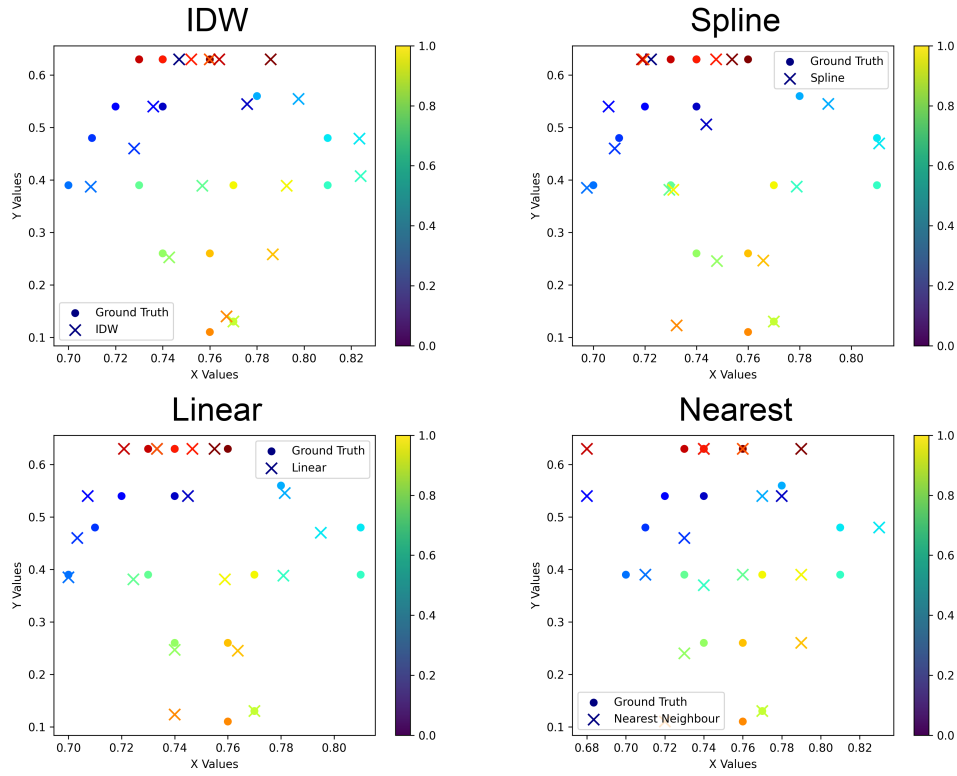


Figure 5.1: Interpolation methods visual joints reconstruction. The circles are the ground truth joints, the cross points are the interpolated ones.

Observing Figure 5.1, it becomes evident that each plot illustrates different behaviors of the algorithms, depending on the underlying mechanisms.

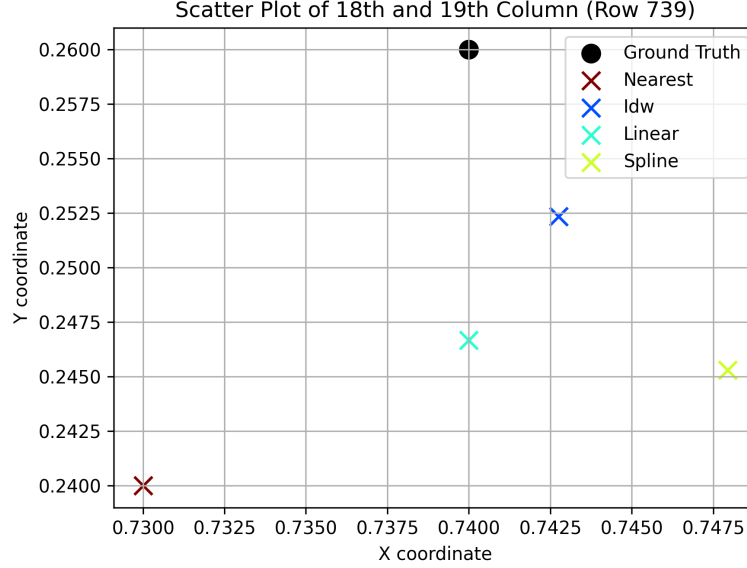


Figure 5.2: Interpolation methods visual joints reconstruction on Right Knee X and Y coordinates.

For a detailed examination of the positional discrepancies among different interpolation methods, a closer inspection can be achieved by zooming in and comparing all interpolations collectively. Fig. 5.2 illustrates the disparity in joint positions for ‘RKneeX’ and ‘RKneeY’ within the same frame as depicted in Fig. 5.1.

As can be noted, Nearest Neighbor is furthest from the ground truth, while IDW and Linear are the closests, indicating their higher effectiveness in approximating the real point coordinates.

5.1.2 Methods Distributions Trends Analysis

In this subsection, we consider all the dataset reconstructed with each method as a distribution. Considering the high quantity of points, we will consider only the distribution of RKneeY and RKneeY.

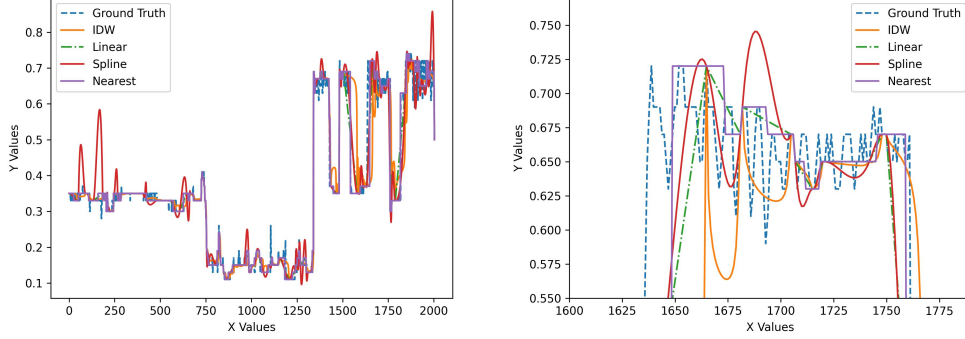


Figure 5.3: Distribution of X and Y coordinates of right wrist. On the left, the overview of the entire distribution counting 2004 values, on the right, a closer look of it.

Illustrated in Fig. 5.3, the Nearest Neighbor method exhibits the poorest behavior, characterized by a step-like trend, this is expected considering the mechanism of the method. On the other hand, Spline displays a similar behavior with smooth curves. The Linear method approximates values without fully capturing both the upward and downward peaks. Lastly, IDW shows steep peaks followed by extended hills.

Next, we explore two additional statistical techniques for analyzing distributions: the ECDF (Empirical Cumulative Distribution Function) and KDE (Kernel Density Estimation). In both methods, the distributions are normalized to ensure their values sum to 1. This normalization is essential as these techniques operate within a probabilistic framework.

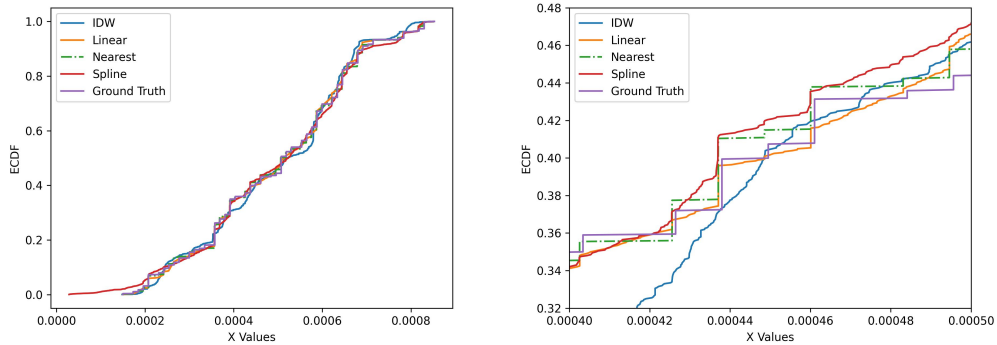


Figure 5.4: Empirical Cumulative Distribution Function of X coordinate of right wrist. On the left, the overview of the entire distribution counting 2004 values, on the right, a closer look of it.

As illustrated in Fig. 5.4, on the left side, we have a plot that represents the entire ECDF, on the right side we have a zoomed-in version of the ECDF, focusing on a specific segment ranging approximately from 0.32 to 0.48.

From the plots, it can be observed that the Spline and IDW interpolations appear to provide smoother transitions compared to others like Nearest Neighbor and ground truth, which show a step-like pattern. Inverse Distance Weight has the lowest of all values, while Spline has the highest.

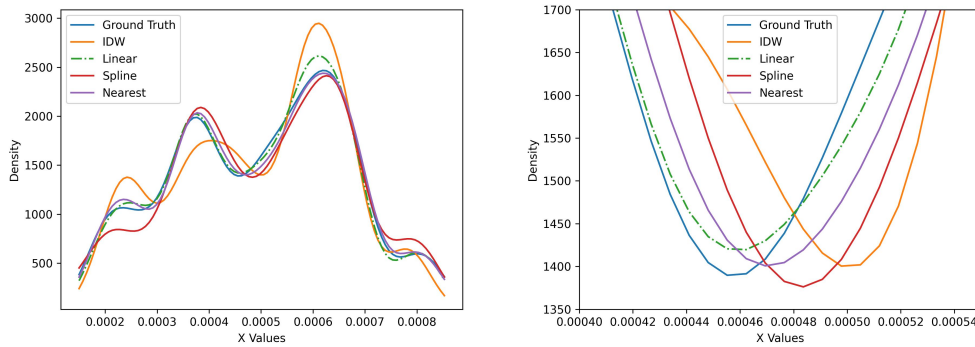


Figure 5.5: Kernel Density Estimation of X coordinate of right wrist. On the left, the overview of the entire distribution counting 2004 values, on the right, a closer look of it.

Even in Fig. 5.5 there are two plots, where the right one is a zoomed-in version of the left one. The left plot represents the entire KDE, with a sharp peak around $X=0.00065$ where all interpolation methods and the ground truth converge. All the interpolation have a smooth curve. The zoomed-in version of the KDE on the right, focuses on a specific segment near $X=0.00046$. All curves follow a similar trend but with slight variations in curvature and steepness. The curve closest to the ground truth is the Linear one. Due to the diverse nature of algorithm, some interpolation methods such as linear, Inverse Distance Weight or Nearest Neighbor approximate the curve having values contained in the area over the curve. On other hand, spline interpolation exceed the local minimum. This is because cubic splines use piecewise cubic polynomials to fit the data, and these polynomials can have local extrema between the given data points.

In cubic spline interpolation, the polynomial segments between data points are not restricted to staying within the range of the given values.

Chapter 6

Conclusions

In this work, we focused on estimating missing joints in the human body using different interpolation methods.

First, we took a dataset from Kaggle that was suitable for our task. The dataset considered consists of four actions and less than three thousand frames. It was suitable for our task because of its deterministic nature.

We performed some pre-processing operations on the dataset in order to have some ground truth values to compare with. So we extracted a subset of the dataset, looking at 7 contiguous rows with no zero values, and created zero sequences for each column.

Then we proceeded to evaluating the effectiveness of different interpolation techniques on these sequences, compared to the ground truth values. The evaluation metrics considered were listed and the results obtained were properly discussed. From the experiments, both quantitative and qualitative, linear interpolation was found to have a higher performance than others. The trend of all distributions of values per joint has been shown in appropriate figures.

The code for the main parts of the project is available at <https://github.com/Lilith-E/BPEw-interpolations/tree/main>.