# Project 3: Robot Understanding of Human Behaviors Using Skeleton-Based Representations CSCI-573 Human-Centered Robotics

Casey Duncan
Colorado School of Mines
1500 Illinois St, Golden, CO 80401
caseyduncan@mymail.mines.edu

***Abstract*— The ability to have robots understand human behaviors is a popular area of research in the field of human centered robotics because it not only allows robots to understand human intent, but also learn how humans interact with the world. A popular method to teach robots how to understand human behavior is by tracking the positions of human skeletal joints and extracting dimensional features from them. In this paper, I will discuss the popular MSR Daily Activity 3D dataset used to train robots in understanding human behaviors. Additionally, I discuss how to use this data to extract dimensional features using the Relative Angles and Distances (RAD), Histogram of Joint Position Differences (HJPD), and Histogram of Oriented Displacements (HOD) representation algorithms. Afterward, I explain the purpose of Support Vector Machines (SVMs), how input data is mapped into higher dimensional spaces, and how to use the features extracted from the representation algorithms to train and test Support Vector Machines (SVMs). Finally, I discuss the results of each algorithm, why the HOD algorithm performs better than HJPD & RAD, and why the HJPD algorithm performs better than RAD. Following the procedure outlined in this paper will allow reader to train a model capable of understanding human behaviors with an accuracy of 60.4% using RAD, 83.3% using HJPD, and 89.6% using HOD.**

## I. INTRODUCTION

The purpose of this project was to implement several skeleton-based representations with Support Vector Machines (SVMs) to classify multiple different human behaviors using a public activity dataset collected from a Kinect V1 sensor. The learning human behaviors are shown in Figure 2 and the skeleton-based representations used for this projects are as follows:

- Relative Angles and Distances (RAD)
- Histogram of Joint Position Differences (HJPD) [1]
- Histogram of Oriented Displacements (HOD) [2]

Details on how these representations are made and used with SVMs will be discussed in section II. Following how they are implemented, I discuss how each representation performs and which one is the most accurate in classifying human behaviors when training an SVM. This entire project was programmed in the Python programming language and implements the LIBSVM[1] third party library for SVM training & testing.

[1]The LIBSVM library information, guide, and download link can be found at https://www.csie.ntu.edu.tw/∼cjlin/libsvm/

## II. APPROACH

The first step toward solving this problem involved gathering and understanding the data. Fortunately, the data was given to me and quite easy to understand. Next, I needed to convert the raw data into each of the representation formats listed in section I. Then, I trained an SVM model for each representation and tuned each model's hyperparameters to find which hyperparameters allowed the model to perform best on the testing data. Instructions on how to execute these steps are outlined in sections II-B - II-D.

### A. DATASET

The skeleton joint data used to construct each representation is from the MSR Daily Activity 3D dataset[2], which was one of the most widely applied benchmark datasets in human behavior understanding tasks. The dataset consists of 16 human activities, as shown in Figure 2, performed by 10 different people. If possible, each person performs all activities twice: first in a standing position and second while sitting.

This dataset contains both color-depth & skeleton joint position data, but for this project I only used the skeleton joint position data to construct the skeleton-based human representations. The skeletal data in each frame contains 20 joints, as shown in Figure 1, and the skeletal joint position data consists of the position (x,y,z) of each joint in each frame.

The skeleton data files used in the project have been preformatted and contain a subset of the original dataset. Only 6 of the total 16 activity categories were used from the dataset and are listed below:

- Cheer Up (a08)
- Toss Paper (a10)
- Lie on Sofa (a12)
- Walk (a13)
- Stand Up (a15)
- Sit Down (a16)

This dataset also contains both training and testing files, with 60% (72 files) of the data making up training data and 40% (48 files) of the data making up testing data. All instances

[2]The MSR Daily Activity 3D dataset has been removed from the author's website and no longer publicly available after 2017.

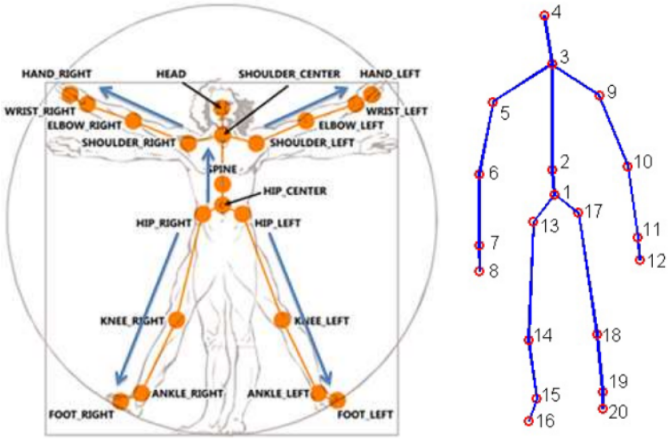


Fig. 1. Skeleton Joint Names (left) and Indices (right)

from human subjects 1-6 are used for training and all instances from subjects 7-10 are used for testing. File names are formatted by activity number, human subject number, and the human subject number's trial number. An example of this looks like: `a08_s01_e01_skeleton_proj.txt` with `a08` being activity number 08, `s01` being subject number 01, and `e01` being trial number 01 for the human subject. All activity numbers are shown in Figure 2. Within each file the data is listed in the ordered format of frame number, joint number, x-position, y-position, & z-position. An example of a single data instance for a single joint within a single frame looks like: `2 4 -0.025 0.214 2.259`, with `2` being the frame number, `4` being the joint index number, `-0.025` being the x-position, `0.214` being the y-position, and `2.259` being the z-position. All joint index numbers are shown in Figure 1.

### B. REPRESENTATION CONSTRUCTION

As stated above, the skeleton-based representations used for this projects are as follows:

- Relative Angles and Distances (RAD)
- Histogram of Joint Position Differences (HJPD) [1]
- Histogram of Oriented Displacements (HOD) [2]

Using the data files detailed in section II-A, each of these representations extracts features from the skeletal joint position data and puts them into a histogram format, counting how many features appear in each bin. For this project, the bin size is a hyperparameter that is tuned by the user. Once in a histogram format, the histogram for each input data file is converted into the SVM format required for LIBSVM and saved into an output file. Implementation of each of these representations is as follows.

*1) RAD:* Unlike the other representations, the RAD representation does not use the data for all 20 skeletal joints. Instead, it uses a single reference joint & the five skeletal joints that form a star, which are the boxed joints shown in Figure 3. The reference joint is ideally in the center of the skeleton, so I chose to use the joint at the center of the hip (joint 1). The other joints used are the head (joint 4), right hand (joint 8), left hand (joint 12), right foot (joint 16), and the left foot (joint 20). Using these joint positions, RAD requires that the euclidean distance $d$ between each star joint and the reference joint be calculated, as well as the angle $\theta$ between the vector connecting the reference joint to a star joint and it's neighboring vectors. Reference Figure 3 for a figure of $d$ & $\theta$. In total, there are 5 distances and 5 angles calculated for each frame. Using these values, a histogram is generated for each $d$ & $\theta$ value for each data file, with the histogram range being the minimum and maximum of each $d$ & $\theta$ value from all input files and the bin size being defined by the user. Once completed, the histograms holding all feature values for all input data files are saved to output files (`rad_d2` for training and `rad_d2.t` for testing) in the LIBSVM format. The algorithm for this process is provided in algorithm 1.

**Algorithm 1** RAD representation using star skeletons

**Input 1:** Training or Testing files from section II-A.
**Input 2:** User selected histogram bin size $b$.
**Output:** `rad_d2` or `rad_d2.t`

```
for each input file do
    for each video frame within file do
        Select star skeleton joints.
        Save (x,y,z) star skeleton joint positions.
        Compute distance d from star joints to ref joint.
        Compute angle θ between each star joint vector.
        Append d & θ to matrix of values for each frame.
    end for
    Find min & max d & θ for each type in each file.
end for
Find min & max of each d & θ type in all files.
Histogram ranges will be defined by min & max values.
for each input file do
    Save each θ type in histogram using range & b.
    Save each d type in histogram using range & b.
    Normalize histograms based on frame quantity.
    Combine all 10 histograms into single histogram.
    Save histogram with input file activity number.
end for
Save all histograms into output file.
```

*2) HJPD:* Unlike the RAD representation, the HJPD representation requires all 20 skeletal joint positions to be used. For HJPD, the features are defined as the joint displacement between a given joint location (x,y,z) and a reference joint ($x_c$, $y_c$, $z_c$), as shown in equation 1.

$$(\Delta x, \Delta y, \Delta z) = (x, y, z) - (x_c, y_c, z_c) \quad (1)$$

The reference joint is ideally in the center of the skeleton, so I chose to use the joint at the center of the hip (joint 1). Because there are 20 joints total and three features ($\Delta x$, $\Delta y$, & $\Delta z$) per joint pair (joint, reference joint), there will be 57 measurements per frame. Once all three measurements are calculated for all frames and all input files, each of the

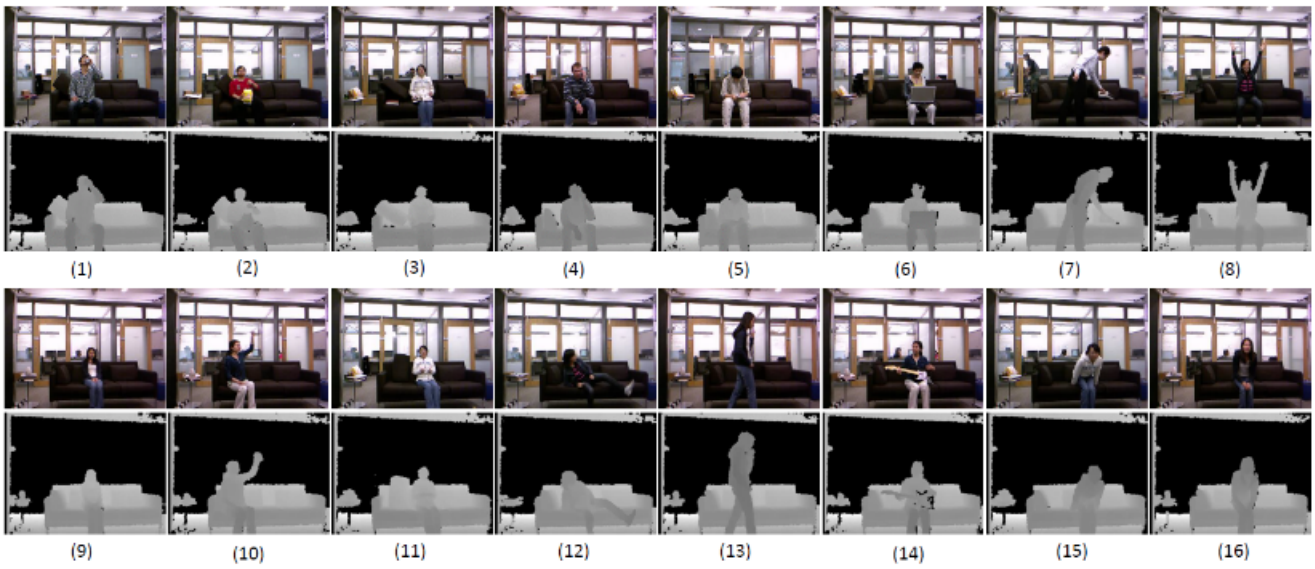


Fig. 2. The full MSR Daily Activity 3D dataset contains sixteen human activities: (1) drink, (2) eat, (3) read book, (4) call cell phone, (5) write on a paper, (6) use laptop, (7) use vacuum cleaner, (8) cheer up, (9) sit still, (10) toss paper, (11) play game, (12) lie down on sofa, (13) walk, (14) play guitar, (15) stand up, (16) sit down

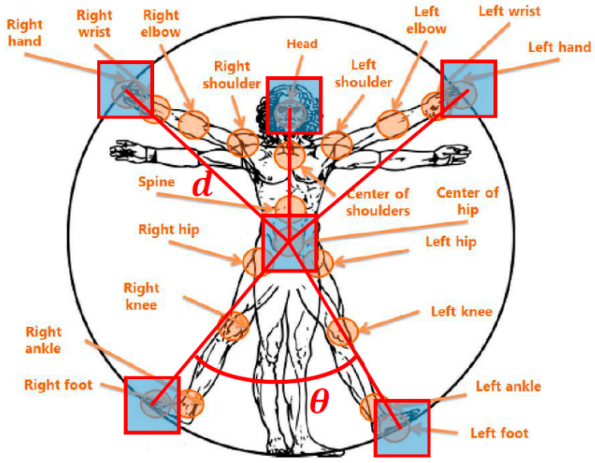


Fig. 3. RAD Star Joints and Features $d$ & $\theta$

57 measurement types are saved into their own histogram for every file. The histogram's range is the minimum and maximum of each measurement type from all input files and the bin size is defined by the user. Next, the histograms for each 3 joint pair measurements $(\Delta x, \Delta y, \Delta z)$ are combined into a single histogram, so there are 19 histograms (one for each joint pair) for each file. Then, all 19 histograms are appended together for each file. Once completed, the histograms holding all feature values for all input data files are saved to output files (`hjpd_d2` for training and `hjpd_d2.t` for testing) in the LIBSVM format. The algorithm for this process is provided in algorithm 2.

*3) HOD:* Similar to the HJPD representation, the HOD representation requires all 20 skeletal joint positions to be used. For HOD, each joint is projected from a 3D trajectory onto the 2D Cartesian planes (i.e., *xy*, *yx*, & *xz*). On each

**Algorithm 2** HJPD representation using all 20 skeletal joints

**Input 1:** Training or Testing files from section II-A.
**Input 2:** User selected histogram bin size $b$.
**Output:** `hjpd_d2` or `hjpd_d2.t`

**for** each input file **do**
  **for** each video frame within file **do**
    Save all 20 (x,y,z) skeleton joint positions.
    Compute $(\Delta x, \Delta y, \Delta z)$ for all joints to ref joint.
    Append $(\Delta x, \Delta y, \Delta z)$ to matrix for each frame.
  **end for**
  Find min & max for each $\Delta x$ type.
  Find min & max for each $\Delta y$ type.
  Find min & max for each $\Delta z$ type.
**end for**
Find min & max of each $\Delta x$ type in all files.
Find min & max of each $\Delta y$ type in all files.
Find min & max of each $\Delta z$ type in all files.
Histogram ranges will be defined by min & max values.
**for** each input file **do**
  Save each $\Delta x$ type in histogram using range & $b$.
  Save each $\Delta y$ type in histogram using range & $b$.
  Save each $\Delta z$ type in histogram using range & $b$.
  Normalize histograms based on frame quantity.
  Combine all $\Delta x, \Delta y, \Delta z$ joint pair histograms.
  Append all 19 histograms together.
  Save histogram with input file activity number.
**end for**
Save all histograms into output file.

2D Cartesian plane, the orientation of the trajectory to the reference coordinate is computed as a HOD feature, which I will call $\theta_{xy}$, $\theta_{yz}$, & $\theta_{xz}$. To calculate these features, the vector between a given joint position in frame $i$ $(x_i, y_i, z_i)$ and the subsequent frame $(x_{i+1}, y_{i+1}, z_{i+1})$ is calculated as shown in equation 2.

$$(\Delta x, \Delta y, \Delta z) = (x_i, y_i, z_i) - (x_{i+1}, y_{i+1}, z_{i+1}) \tag{2}$$

Next, the angles $\theta_{xy}$, $\theta_{yz}$, & $\theta_{xz}$ in the principle plane projections for each joint are calculated as shown in equation 3.

$$\begin{aligned} \theta_{xy} &= \text{atan2}(\Delta y, \Delta x) \\ \theta_{yz} &= \text{atan2}(\Delta z, \Delta y) \\ \theta_{xz} &= \text{atan2}(\Delta z, \Delta x) \end{aligned} \tag{3}$$

There will be a total of 60 different features per frame pair (20 joints and 3 angles per joint). Next, the following 7 histograms are created:

- Histogram capturing data in all frames
- Histogram capturing data in first half of all frames
- Histogram capturing data in second half of all frames
- Histogram capturing data in first quarter of all frames
- Histogram capturing data in second quarter of all frames
- Histogram capturing data in third quarter of all frames
- Histogram capturing data in fourth quarter of all frames

When creating these histograms, the histogram range is from $-\pi$ to $+\pi$ and the bin size is selected by the user. Next, the 7 histograms are combined into a single histogram, resulting in one histogram for each file. Once completed, the histograms holding all feature values for all input data files are saved to output files (`hod_d2` for training and `hod_d2.t` for testing) in LIBSVM format. The algorithm for this process is provided in algorithm 3.

**Algorithm 3** HOD representation using all 20 skeletal joints

**Input 1:** Training or Testing files from section II-A.
**Input 2:** User selected histogram bin size $b$.
**Output:** `hod_d2` or `hod_d2.t`

**for** each input file **do**
  **for** each video frame within file **do**
    Save all 20 (x,y,z) skeleton joint positions.
    Compute $(\Delta x, \Delta y, \Delta z)$ for all joints between frame $i$ and $i+1$.
    Compute $\theta_{xy}$, $\theta_{yz}$, & $\theta_{xz}$ for all joints.
    Save all 60 angle types for each frame pair.
  **end for**
  Create 7 histograms for each of the 60 angle types.
  Normalize histograms based on frame quantity.
  Combine all 7 histograms into one.
  Save histogram with input file activity number.
**end for**
Save all histograms into output file.

### C. Support Vector Machines (SVMs)

SVMs are a machine learning method that can define a function for a line or hyperplane, which separates two or more categories of data. Therefore, when the input data is plugged into the hyperplane function, its category is determined by which side of the line(s), or plane(s), the input data falls on. The equation for this plane is $f(x) = w \cdot x + b \geq 1$ where $\frac{1}{\|w\|}$ represents the distance from the hyperplane to the closest points of each class that the hyperplane is separating. It should be noted that each of these points lie on a plane that is parallel to the hyperplane called the support plane. Therefore, the SVM optimization problem becomes as is shown in equation 4.

$$\begin{aligned} &\min \|w\| \\ &\text{s.t. } y_i(w \cdot x + b) \geq 1 \end{aligned} \tag{4}$$

Generally, a hyperplane that separates each category cannot be constructed since the data sometimes falls into the clusters of other classes. As a result, a variable to capture error must be added and the SVM optimization problem becomes as is shown in equation 5.

$$\begin{aligned} &\min \|w\| + C\sum_{i=1}^{l} \zeta_i \\ &\text{s.t. } y_i(w \cdot x + b) \geq 1 - \zeta_i \\ &\quad \zeta_i \geq 0 \end{aligned} \tag{5}$$

In equation 6, the variable $\zeta_i$ represents the distance between the data point that falls outside of its correct class to the support plane of its correct class. Also, $C$ is a hyperparameter that allows the user to choose how much these outlier points have an effect on the optimization problem.

Using the output training and testing representation files generated in section II-B, I was able to train and test SVM models for each of the representations using the popular open source LIBSVM library. In most cases, training and testing data is non-linear and needs to be projected into a higher dimensional space in order to be separable by a hyperplane. To do this, the data $(x)$ is mapped into the higher dimensional space using the function $\phi(x)$. Using this function allows for the SVM optimization problem to change to what is shown in equation 6.

$$\begin{aligned} &\min \|w\| + C\sum_{i=1}^{l} \zeta_i \\ &\text{s.t. } y_i(w \cdot \phi(x) + b) \geq 1 - \zeta_i \\ &\quad \zeta_i \geq 0 \end{aligned} \tag{6}$$

Because the dimensionality of $\phi(x)$ can be very large, it is difficult represent $w$ in the computers memory, thus making the optimization problem hard to solve. Therefore, the Representer theorem is used to optimize some variables $\alpha$ instead of directly optimizing $w$. This theorem is shown in equation 7.

$$w = \sum_{i=1}^{m} \alpha_i \phi(x_i) \tag{7}$$

This changes the SVM optimization problem constraints to be as shown in equation 8.

$$\text{s.t. } y_i(\sum_{i=1}^{m} \alpha_i \phi(x_i) \cdot \phi(x) + b) \geq 1 - \zeta_i \qquad \zeta_i \geq 0 \tag{8}$$

Here, $\phi(x_i) \cdot \phi(x)$ is called the Kernel Function, $K(x_i, x)$. There are several different types of Kernel Functions, such as the linear, polynomial, radial basis function (RBF), or sigmoid. For this project, I used the RBF function, which is shown in equation 9.

$$K(x_i, x) = \phi(x_i) \cdot \phi(x) = \exp(-\gamma \|x_i - x\|) \tag{9}$$

In this equation, $\gamma$ is another hyperparameter that the user can change to improve the accuracy of the solution. The LIBSVM library allows the user to easily select which Kernel Function they want to use, as well as the optimization problems and Kernel Function's hyperparameters. Therefore, training and testing a model with the files generated in section II-B only requires a couple lines of Python code.

### D. Training

Based on information provided in sections II-B and II-C, I had the following hyperparameters that I was capable of using to tune my SVM models in producing the most accurate model for each skeleton representation:

- RAD angle histogram bin size, $b_{rad-\theta}$
- RAD distance histogram bin size, $b_{rad-d}$
- HJPD distance histogram bin size, $b_{hjpd}$
- HOD distance histogram bin size, $b_{hod}$
- The $C$-value within equation 6
- The $\gamma$-value within equation 9

To tune the bin sizes, I created a command line interface (CLI) that allowed me to enter a desired bin size for each bin size hyperparameter from the terminal. Please reference the `README.md` file within my code for how to use the CLI. For the $C$ and $\gamma$ values, the LIMSVM library provides a program called `grid.py` that performs a grid search for values of $C$ and $\gamma$ within a user defined range. I chose to use the default range, which is between $2^{-5}$ & $2^{15}$ for $C$ and between $2^{-15}$ & $2^{4}$ for $\gamma$. The output of this grid search is the best accuracy for the tested $C$ and $\gamma$ values, along with the $C$ and $\gamma$ values that produced that accuracy. An example of this output is shown in Figure 4. Here, it shows that this was a grid search performed on the HJPD training data histogram file, `hjpd_d2`, the best accuracy is 80.5556%, and the $C$ & $\gamma$ values that produce this accuracy are $C = 8$ & $\gamma = 0.125$.

## III. EXPERIMENTS & RESULTS

Using the CLI described in section II-D, I used the training data to train models with all bin sizes ($b_{rad-\theta}$, $b_{rad-d}$, $b_{hjpd}$, & $b_{hod}$) ranging from 10 to 25. A plot of the bin size versus the best grid search accuracy percentage for each representation is shown in Figure 5. Using this information, I found that the following bin size values produced the best grid search accuracies for the given representations:

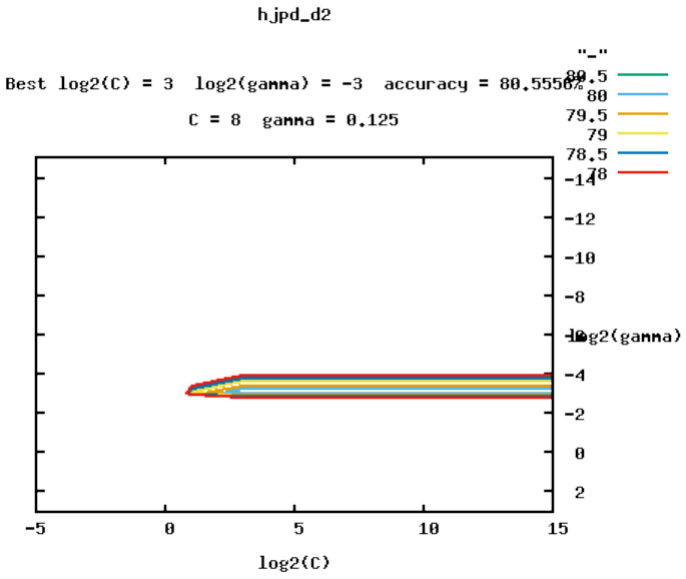


Fig. 4. LIBSMV Grid Search Output Example

- RAD: $b_{rad-\theta} = b_{rad-d} = 11$
- HJPD: $b_{hjpd} = 11, 18,$ & $24$
- HOD: $b_{hod} = 22$

Because there were three different bin sizes that all produced the best accuracy for HJPD, I tested each with the $C$ and $\gamma$ values provided by each of their grid searches to find the one that produced the best accuracy on the testing data. Afterwards, I found that the optimal bin size for HJPD was 11.

Using the grid search output for the bin sizes listed above, I was able to find the optimal $C$ and $\gamma$ values for each representation. The values for each hyperparameter and representation testing data accuracies is shown in Table I and the confusion matrix for each representation can be seen in Tables II - IV. When reading the confusion matrices, please note that the rows represent the true values and the columns represent the predicted values.

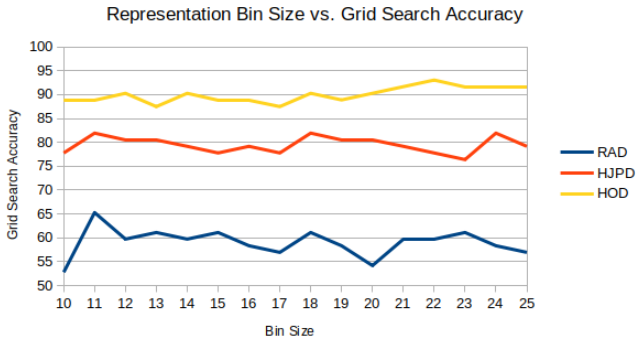


Fig. 5. Representation Bin Size vs. Best Grid Search Accuracy

TABLE I

OPTIMIZED REPRESENTATION HYPERPARAMETERS VALUES & TEST DATA ACCURACY

| Representation | Bin Size | $C$ | $\gamma$ | Test Accuracy |
|---|---|---|---|---|
| RAD | $b_{rad-\theta} = b_{rad-d} = 11$ | 8 | 0.03125 | 60.4167% |
| HJPD | $b_{hjpd} = 11$ | 2 | 0.125 | 83.333% |
| HOD | $b_{hod} = 22$ | 2 | 0.02125 | 89.5833% |

TABLE II

RAD BEST TEST ACCURACY CONFUSION MATRIX

| Activity No. | 08 | 10 | 12 | 13 | 15 | 16 |
|---|---|---|---|---|---|---|
| 08 | 8 | 0 | 0 | 0 | 0 | 0 |
| 10 | 0 | 4 | 2 | 0 | 1 | 1 |
| 12 | 0 | 0 | 4 | 2 | 1 | 1 |
| 13 | 0 | 1 | 1 | 3 | 2 | 1 |
| 15 | 0 | 0 | 0 | 0 | 5 | 3 |
| 16 | 0 | 0 | 0 | 0 | 3 | 5 |

TABLE III

HJPD BEST TEST ACCURACY CONFUSION MATRIX

| Activity No. | 08 | 10 | 12 | 13 | 15 | 16 |
|---|---|---|---|---|---|---|
| 08 | 8 | 0 | 0 | 0 | 0 | 0 |
| 10 | 0 | 8 | 0 | 0 | 0 | 0 |
| 12 | 0 | 0 | 7 | 0 | 0 | 1 |
| 13 | 0 | 2 | 0 | 6 | 0 | 0 |
| 15 | 0 | 0 | 0 | 0 | 4 | 4 |
| 16 | 0 | 1 | 0 | 0 | 0 | 7 |

TABLE IV

HOD BEST TEST ACCURACY CONFUSION MATRIX

| Activity No. | 08 | 10 | 12 | 13 | 15 | 16 |
|---|---|---|---|---|---|---|
| 08 | 8 | 0 | 0 | 0 | 0 | 0 |
| 10 | 1 | 7 | 0 | 0 | 0 | 0 |
| 12 | 0 | 0 | 8 | 0 | 0 | 0 |
| 13 | 0 | 2 | 0 | 6 | 0 | 0 |
| 15 | 0 | 0 | 0 | 0 | 8 | 0 |
| 16 | 0 | 2 | 0 | 0 | 0 | 6 |

## IV. DISCUSSION

Looking at the confusion matrices shown in Tables II - IV, it can be seen that the overall best classified human activity is Cheer Up (a08). The worst performing human activity is Walk (a13), which was mostly confused with Toss Paper (a10). Based on Figure 5, it can easily be seen that the HOD representation performs the best when training an SVM to classify the chosen human behaviors listed in section II-A, HJPD is a close second, and RAD being the least accurate of the three. After reviewing each algorithm, I believe that HOD performs with the highest accuracy because it is the only algorithm where the output feature is tracking the movement of each joint between frames, which allows it to be speed-invariant & scale-invarient. Alternatively, HJPD and RAD are simply talking the position of each joint with respect to a reference joint in a single frame. Another advantage to the HOD algorithm is that it applies a temporal pyramid (3-level pyramid: all, half, & quarters of data) to capture the joint trajectories, which includes more temporal information. On the other hand, HJPD and RAD only form a single histogram for each frame.

Comparing the algorithm's for HJPD and RAD, it appears that HJPD is the better of the two because it measures the distances from all joints to a single reference joint, while RAD only compares 5 of the 20 joints to a reference joint. In other words, HJPD (57 features per frame) produces almost six times as many features per frame as RAD (10 features per frame). Also, HJPD supplies the required information to extract additional features, such as Joint Movement Volume Features, which are the bounding volumes that each joint moves within. On the other hand, no additional features can be extracted from RAD.

In summary, all three representations are sufficient in teaching a robot to understand human behaviors at an accuracy of 60% or greater. However, HOD and HJPD are far superior than RAD, with HOD producing the best accuracy of all three.

## REFERENCES


[1] Hossein Rahmani, Arif Mahmood, Du Q Huynh, and Ajmal Mian. Real time action recognition using histograms of depth gradients and random decision forests. In *IEEE winter conference on applications of computer vision*, pages 626–633. IEEE, 2014.

[2] Mohammad Abdelaziz Gowayyed, Marwan Torki, Mohammed Elsayed Hussein, and Motaz El-Saban. Histogram of oriented displacements (hod): Describing trajectories of human joints for action recognition. In *Twenty-Third International Joint Conference on Artificial Intelligence*, 2013.