2013 IEEE/RSJ International Conference on
Intelligent Robots and Systems (IROS)
November 3-7, 2013. Tokyo, Japan

# Recognition of Ballet Micro-Movements for Use in Choreography

Justin Dancs, Ravishankar Sivalingam, Guruprasad Somasundaram, Vassilios Morellas, and Nikolaos Papanikolopoulos
dancs003@me.umn.edu, ravi@cs.umn.edu, guru@cs.umn.edu, morellas@cs.umn.edu, npapas@cs.umn.edu
Department of Mechanical Engineering, Department of Computer Science
University of Minnesota
Minneapolis, MN 55455

*Abstract*— **Computer vision as an entire field has a wide and diverse range of applications. The specific application for this project was in the realm of dance, notably ballet and choreography. This project was proof-of-concept for a choreography assistance tool used to recognize and record dance movements demonstrated by a choreographer. Keeping the commercial arena in mind, the Kinect from Microsoft was chosen as the imaging hardware, and a pilot set chosen to verify recognition feasibility. Before implementing a classifier, all training and test data was transformed to a more applicable representation scheme to only pass the important aspects to the classifier to distinguish moves for the pilot set. In addition, several classification algorithms using the Nearest Neighbor (NN) and Support Vector Machine (SVM) methods were tested and compared from a single dictionary as well as on several different subjects. The results were promising given the framework of the project, and several new expansions of this work are proposed.**

## I. Introduction

Computer vision applications are wide and diverse. Potential uses have been found from security to manufacturing to even entertainment. As more hardware technology and software algorithms are developed, applications for this new technology will expand into new and different areas. One such area is dance recognition, specifically for use in ballet choreography.

This paper explores the realm of ballet and dance and the use of computer vision to enrich the field, using the Kinect from Microsoft to segment and track joint coordinates of the body. Prior work in dance recognition was explored and summarized below. Also, a contextual overview of ballet and choreography is provided, as well as the motivation and vision that began this work. The representation scheme and the reasoning for it are then explained. Several methods of classification are then compared, in both Nearest Neighbor (NN) methods and Support Vector Machine (SVM) methods, discussing the results of testing cross-validation as well as new, different dancers on a trained library with a different dancing set. Finally, future expansions of this project are proposed, both improving the current classification scheme as well as expanding its capabilities.

## II. Related Works

According to Aggarwal et. al, computer vision applications fall into three broad categories: 1) motion analysis segmenting and identifying a human body, 2) tracking human motion, and 3) activity recognition of human motion [1]. The API from the Kinect handles the first two categories in this work, as the constraints of the Kinect are not a problem in the context of ballet; the third task of activity recognition was the primary objective to explore in this project.

There has been some previous work in the form of dance recognition, although differing in one or more aspects to the work detailed in this paper. Deng et al. used a form of Segmented Single Value Decomposition, dubbed, SegSVD, to recognize simple movements of varying length of *a-go-go* dance movements [2]. Principal component analysis (PCA) including functions like SVD is one approach to recognition, simplifying the segments into a smaller set of important features to use. Methods like PCA and SVD can be most commonly used when multiple points on the body are tracked and used for recognition to reduce to a lower number of more significant features. In contrast, another method of recognition is using a silhouette of a subject to recognize movements, implementing a "tunneling" system that compares the silhouette to a known silhouette configuration of a recognizable move. A 3D example of this can be found in the work of Peng et al. at Arizona State University [3] [4].

Most notable in prior art is the work of Lee Campbell and Aaron Bobick at MIT in 1995 [5], who attempted a very similar project to the one described in this paper. Their approach used angles and orientations assuming the cartesian coordinates of joints, as well as a similar approach to breaking ballet movements into segments. Their work used the Multi-Trax system, which was a commercially available system for 3D tracking at the time. This system used a marker physically on the body, and was an expensive system of capture.

Others have also used recognition and classification algorithms similar to the work described in this project, for differing reasons. Using PCA, Tsuruta et al. [6] used a NN classifier to classify dance movements for a "dance collaboration system" utilizing 12 cameras to capture the subject. Although the algorithm is similar to algorithms presented in our paper, the Kinect offers a more feasible advantage for common use as it requires less setup and is more cost effective to implement.

A large part of the action recognition methodology in this work most closely resembles the work of Sivalingam et al. out of the same lab as this work [7]. When trying a NN algorithm and comparing results on segmented data from the Kinect and the accompanying Application Programming Interface (API), it was found that the trained classifiers produced satisfactory results without any additional, more complicated or computationally expensive algorithm. SVM was also tried, and achieved moderate results given the training set provided in this introductory work, which uses known algorithms and classification methods on a new area understudied in computer vision.

## III. Ballet

According to the National Dance Education Organization, there are approximately 32,000 dance studios currently in the US, and 3.5 million children are currently taught dance by professionals. In addition, there are 665 institutions that offer dance major and minor programs in the US [8].

At the onset of this project, the eventual vision, and therefore scope, was in choreography. Choreography tools exist on the market currently, like the Dance Designer program offered by ChoreoPro [9]. Systems like this, however, require dance experts to synthesize dances by sitting at a computer and rely on visualization of the program to view and review their work before training subjects to perform their choreography. Dancers are typically kinesthetic learners; that is, they learn by physically moving. Thus, choreography is typically created by performing the dance themselves. In observing dancers synthesizing choreography, it was noticed that most approach the challenge of documentation in one of two ways. One approach is to dance out a short segment, then stop and document the segment by writing it down, then dance out the next segment. This process is repeated until the final product has been created. However, a person's memory is limited, so the need for intermittent stops and starts is disruptive and time-consuming. The other approach to choreography is to video record the dancer performing a dance to disseminate later. This can also be time consuming, as the footage would have to be manually processed later to better document what is shown.

Computer vision algorithms could be used in this application to automatically document the sequential moves of a dancer. This could assist in choreography by recording a dancer's movements to disseminate to others in the future. In addition, this concept could also aid in dance documentation, as certain dance companies perform the same or similar dances repetitively over the years. The goal of this work was to serve as a proof-of-concept and to evaluate the feasibility in varying aspects concerning this idea.

A major aspect of selecting ballet as a prime starting point is in its structure. Compared to other forms of dance, ballet is highly structured and well understood. Dancers and instructors from different areas and different regions of the world have the same basic understanding of each moves and its name with only minor differences. Thus, documenting choreography with names is useful for a wide variety of dancers in this area. In addition, each movement in ballet can be broken into several micro-movements. Each movement starts with a position - typically 1st, 4th, or 5th - and a micro-movement following that position, then expands into other movements that identify the specific step of choreography. Thus, it is possible to organize an extensive library of ballet movements into a decision tree for classification.

This figure organizes the first two decisions in the tree, leaving the third decision as a rapidly expanded set depending on the combination of the first two levels.

The question remained, then, if it was feasible to effectively classify the micro-movements that would comprise each step. Thus, a pilot study was performed to verify that movements in ballet could be effectively classified. The movements chosen were the tendu, the plié, the passé, and the elevé. The plié was also chosen to be in first position and fourth position. Finally, a turn was included in the library to ensure that turns could be classified even with the hardware limitation of the Kinect.

The hardware used in this work is the Kinect by Microsoft, originally released for the Xbox 360 gaming platform. The Kinect and the API in the Kinect Source Development Kit (SDK) is a vision solution to human pose recognition and skeletonization. More information on the Kinect can be found in [10].

## IV. Pre-Processing

As the hardware was used as an out-of-the-box solution for human segmentation and skeletonization, the raw inputs for the action recognition task were 3-dimensional Cartesian coordinates of each of the 20 joints in the skeleton.

A useful feature provided from the Kinect drivers from Microsoft was the introduction of a floor sensing feature, returning a planar equation for the location and angle of the floor. Considering this tool, the assumption was made that any use of this software or algorithm would require the floor to be flat and level. Considering the realm of this project was dance, this is an assumption assumed to be valid for most places for which this concept might be applied. The floor sensing feature was utilized to transform all points of the skeleton from a coordinate system relative to the camera plane to a coordinate system relative to the floor.

Considering the human body from a kinematics perspective, the important information to obtain from the skeletonization is similar to the important variables of a robotic arm. The length of each body segment (arm, leg, hand, neck) are framed by bone and therefore do not vary in length: thus, the important variables that do change is the orientation of each of these segments. Therefore, the coordinates of each joint were converted to 3-dimensional spherical angles for each segment of the body. Since the x, y, and z coordinates of each joint are known, the vector $\langle \delta_x, \delta_y, \delta_z \rangle$ represents the differences of the two connected joints in their x, y and z components, respectively. This is used to create the two angles that represent the 3D orientation of each vector, where $\theta_{joint}$ refers to the polar angle of the vector's projection in the X-Y plane and $\phi_{joint}$ refers to the angle between the vector and the Z axis. Each segment connects two adjacent joints, reducing the 20 joint coordinates into 19 segment orientations representing the same configuration. The two equations for these angles $\theta$ and $\phi$ are shown in Equations (1) and (2). This effectively reduces 60 parameters (20 3-D joint coordinates) into 38 (spherical angles of 19 segments).

$$\theta = \arctan \frac{\delta_y}{\delta_x} \tag{1}$$

$$\phi = \arctan \frac{\delta_z}{\sqrt{\delta_x^2 + \delta_y^2}} \tag{2}$$

After reviewing the atomic moves to be classified, it was noticed that the distinguishing features of each move was the placement and movement of the legs: thus, the only data passed to the classifier was from the lower spine downward. This lowered the number of features from each frame from 38 to 18 (or 19 line segments into 9). Expansion of data may include arm data and arm movements: however, this can be split into separate classification steps if the need is discovered upon expanding the library of moves.

Finally, the goal of the project was action recognition, not pose recognition: therefore, each action was vectorized into one or more feature vectors by taking multiple frames and concatenating them into a longer vector. The final algorithm concatenated 9 frames into a feature vector. Additionally, not all segments were the same size, and few were simply 9 frames long, so a "sliding window" concept was used to concatenate 9 consecutive vectors, slide the window a frame forward (dropping the earliest frame and adding a later one), and concatenate those 9 frames. This was accomplished in MATLAB using the im2colstep function provided in the KSVD toolbox [11]. In this paper, we refer to each individual vector as a segment, and the grouping of vectors from the same action as a sequence.

In this work, each segment provided in the training set and any test sets were cropped manually to include only the significant parts of a dance move. This was done in simulation of a sliding window concept: if a classification algorithm was to run in real-time, not every frame or sampling of frames would be significant. In ballet, there are frames of transition between each dance segment, and the identifying segment of each action is normally near the beginning. As this project is a proof of concept and the algorithm was run off-line, each segment was manually cropped to include just the identifying pieces of a move, by cropping the frames to include just the significant section. This was done to determine whether or not an algorithm could classify a move successfully. If found successful, a future step would be to classify multiple moves in the same segment by determining a minimum correlation. If that correlation threshold was not met, the classifier would skip to the next group of frames.

## V. Classification

This section describes varying methods of classification with varying degrees of success in accuracy. Each algorithm was tested using the same pilot set of training data and classes, using examples from a single dancer performing the specified moves.

The first method tested was basing decisions on the Hausdorff distance. The "voting" nearest neighbor algorithm calculated the distance of each training segment to each test segment, and returns the class of the training segment with the lowest distance for each test segment. With each test segment having a proposed "nearest class", the mode is taken of the resulting classes. The class that has a training segment nearest to the most test segments is determined to be the proposed class of the test sequence. In the event of a tie, the algorithm simply returns the first class. However, in all the results shown, a tie is included as a missed classification. The confusion matrix for the cross validation is shown in Table I. There is no column for "tie," as none were found.

TABLE I
Confusion Matrix of the L1 Voting Nearest Neighbor Algorithm

| | elevé | passé | plié | plié in 4th | tendu | turn in passé |
|---|---|---|---|---|---|---|
| elevé | 15 | 0 | 0 | 0 | 0 | 1 |
| passé | 0 | 18 | 1 | 0 | 0 | 0 |
| plié | 0 | 0 | 11 | 0 | 1 | 0 |
| plié in 4th | 0 | 0 | 0 | 15 | 0 | 0 |
| tendu | 0 | 0 | 1 | 0 | 8 | 0 |
| turn in passé | 0 | 0 | 0 | 0 | 0 | 5 |

One drawback of of the "voting" algorithm was that each segment and distance were equally weighted. One way to mitigate this is to directly use the class and distance, rather than just the nearest class of each segment. This was accomplished using a Radial Basis Function (RBF), where $\sigma$ is a tuned parameter for the training set.

In addition, rather than using the best example of each class, the algorithm uses the distances of all training segments for each class. To this end, the algorithm calculates distances in the same way as the nearest neighbor algorithm; however, it also uses the RBF previously shown on each distance metric to produce a "weight." Once the weight matrix is found, the weights of each class are added together to form sums. This is all performed at a segment level; once the sums are found for each segment, each class's sum for each segment is multiplied together. With this method, if a single segment shows a high sum for a class but every other segment shows a low sum for that same class, the product will be low, and only the class that shows a weighting that is consistently high in each segment will have the highest product and be selected for the proposed class.

To function at the algorithm's best capacity, a sensitivity study was performed using a blind search of parameter $\sigma$ over a range of values. This was done using leave-one-out cross validation as the parameter to optimize. The highest accuracy point was at $\sigma = 0.5$ with an accuracy of 96%, and selected as the optimum value for the use of this algorithm. This accuracy was run using the "cityblock" metric and the Euclidean metric. The Euclidean distance sensitivity study shows a similar trend to the "cityblock" distance, but shifted, with optimum values for $\sigma$ being higher than its counterpart. In future uses of this algorithm, the sigma value is the optimum value from these respective sensitivity functions.

These algorithms were used in the beginning portion of the project. The results of each are plotted in Table II for each method. The accuracy metric refers to the accuracy of data using leave-one-out cross validation on the entire data set, numbering 76 sequences comprising a total of 1,262 segments. This accuracy is classified on the sequence level, so misclassifications of a segment are not recorded if the sequence is still classified accurately. Although the differences are minimal, the later versions of the program experiences a slightly higher accuracy using the L1 norm; this being said, the difference is very minor and would show either version to be comparable.

TABLE II
ACCURACY RESULTS OF NEAREST NEIGHBOR ALGORITHMS USING LEAVE-ONE-OUT CROSS VALIDATION

| Method | Accuracy |
|---|---|
| L1 Hausdorff distance | 2.67% |
| L2 Hausdorff distance | 2.67% |
| L1 voting | 94.67% |
| L2 voting | 93.33% |
| L1 RBF | 96.05% |
| L2 RBF | 94.74% |

The final classifier explored in the realm of this project was the Support Vector Machine, or SVM, algorithm first used by Vapnik [12]. As the SVM algorithm used was a binary classifier, the algorithm recognized more than two classes by employing six separate classification steps and a form of "voting" hierarchy once again to make the final decision. The algorithm actually employs six SVM classification steps attempting to classify the same test segment only as "of a class" or "NOT of a class." These "yes" and "no" votes are tabulated for each segment, and the class that has the most votes in the entire sequences is determined to be the proposed class. Similar to previous algorithms, a tie resulted in the proposition of the first class in order; when this occurred, it was counted as a misclassification in accuracy results.

The results of each kernel are shown in Figure III, once again using leave-one-out cross validation. The linear kernel performed exemplary with the training data provided; as seen in Section IV the data exhibits clusters with small amounts of overlap, thus determining boundaries for classification is less complicated, and the penalizing aspect of the algorithm wouldn't need to be used as much.

Like the previous use of the radial basis function, the value of $\sigma$ needed to be optimized for the radial basis kernel of the training set used. This optimization is shown in Figure 1. In addition, all kernels also optimized for C, the weight of the penalizing function. Interestingly, the value of C affected the accuracy very little. The explanation for this, once again, lies with the separability of the data after the conversion to the

TABLE III
ACCURACY OF KERNEL FUNCTIONS IN SVM USING LEAVE-ONE-OUT CROSS VALIDATION

| Kernel | Accuracy |
|---|---|
| Linear | 98.68% |
| Quadratic | 96.05% |
| Polynomial Order 3 | 90.79% |
| Polynomial Order 4 | 52.63% |
| RBF | 97.37% |

representation scheme; as the data was well separated, few data points would lie on the wrong side of a boundary, and the penalization function would be used very little. Never the less, the optimal value of C was used in each case. The optimal value found for each method was very small, with the exception of the RBF kernel. As the RBF is effectively an inverse function, this is to be expected. The box constraint used for every other kernel was 0.03125, or $2^{-5}$, with the box constraint for the RBF function serving as 0.5, although any value above 0.125 produced similar results.

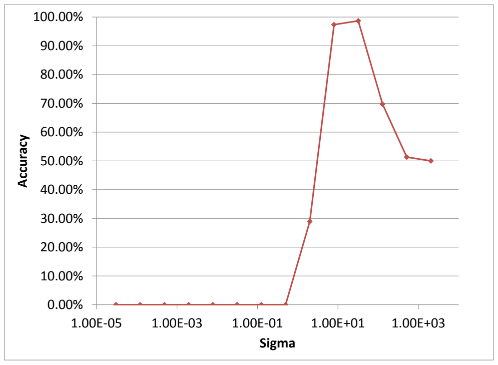

Fig. 1. Sensitivity study of the radial basis function kernel in SVM.

All classification performances shown up to this point have been leave-one-out cross validation with a dictionary of a single dancing subject. However, unknown factors can change when attempting to classify motions of another dancer. Differences such as style of movement, speed, and even unforeseen factors in their motions or ambient conditions can produce undesirable results. Therefore, the above described algorithms were tested using different dancers. The dancers all were experienced in varying degrees with ballet, and knew how to do the moves correctly. However, the dancers performed the moves in the set without significant coaching on technique or motion, and only the explanation of what move to perform was provided. Each test segment was manually cropped into their individual identifying actions in a manner consistent with the processing of the training data as explained in Section IV.

The results for each algorithm across the entire set of dancers is shown in Figure 2. There are several factors attributing to these accuracy values. In keeping with a "hands-off" approach to coaching the dancers, the moves were named and only simple information was given to them to ensure the correct move. Some dancers used a different leg than the dancer used in training; thus, each of those moves was misclassified. A fix to that problem is easily accomplished with a more complete library, and would need to be implemented for a more robust solution. In addition, certain moves were interpreted vastly different than others: the moves described in this pilot set were known, but combinations of these moves were named differently, thus referring to them as the sum of their parts was occasionally unfamiliar to the subjects. However, the data is useful especially as a comparison of methods and a test to the algorithm's ability to classify dance movements with unforeseen variations.

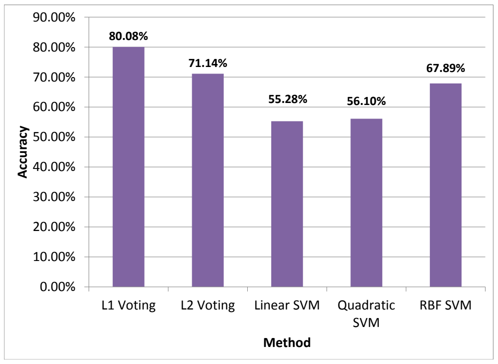

Fig. 2. Accuracy comparison of methods using different dancing subjects.

The variations of accuracy over the methods were shown to scale for each method according to the dancer involved. A plot of the average accuracy of each dancer is shown in Figure 3. The maximum, minimum, and average accuracy are plotted. They roughly vary proportionally from dancer to dancer; some dancers performed well in comparison to the training set across all algorithms, others not as well. It was noted that the two lowest dancers, 3 and 5, both did movements fast, and the member that did movements the slowest, 4, has the highest consistent accuracy. It is therefore theorized that a major part of this discrepancy is the speed at which movements are made. This would warrant a time warping aspect of future versions of this classifier.

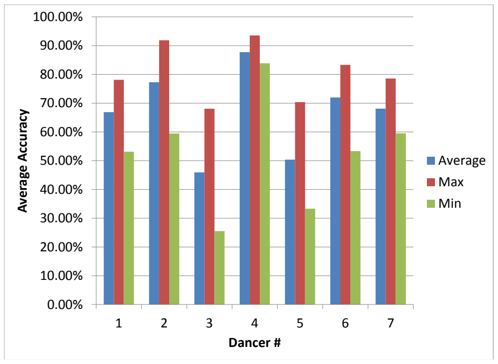

Fig. 3. Average, minimum, and maximum accuracy for each dancer.

A confusion matrix for the voting algorithm using the L1 norm is shown in Table IV, being the strongest algorithm in both cross-validation and new subject testing. Some of the largest misclassifications were the plié in 4th position being classified as a plié in 1st position or as a tendu. This is intuitive, as a plié in 4th position has one foot out in front of the body like a tendu, but still exhibits the motions of a plié. In addition, out of all the classes, the plié in 4th position was least represented in the training data, having only two source sequences that involved it.

TABLE IV
CONFUSION MATRIX OF THE L1 VOTING NEAREST NEIGHBOR ALGORITHM

| | elevé | passé | plié | plié in 4th | tendu | turn in passé | tie |
|---|---|---|---|---|---|---|---|
| elevé | 39 | 0 | 0 | 0 | 4 | 0 | 2 |
| passé | 1 | 64 | 0 | 0 | 1 | 0 | 0 |
| plié | 0 | 0 | 43 | 1 | 1 | 0 | 1 |
| plié in 4th | 0 | 0 | 12 | 4 | 4 | 1 | 1 |
| tendu | 0 | 7 | 6 | 0 | 35 | 4 | 0 |
| turn in passé | 1 | 1 | 1 | 0 | 0 | 12 | 0 |

## VI. FUTURE WORK/LESSONS LEARNED

There are several areas that could expand from the initial work done in this area. When visualizing the future goals and path that a project should take, care must always be taken to keep the final project in mind. With this project, the overall goal was to move towards the development of a recognition tool for choreography. Thus, the program needs to be on-line and easy for dancers to use effectively for recording the moves in a dance.

In addition to other areas there are also several tasks that can be done to bring this initial work to that goal, and additional features that could be tied in to assist this vision. First of all, the program needs to be brought on-line and optimized. This will require the ability to do a recognition task faster than 30 Hz if it was required to use every frame of skeleton data returned by the Kinect API. Although the algorithm described in this work is fast, any additions must keep processing speed in mind as a design factor.

Minor advances could also be made in the methodology of the classification and calibration programs. The Kinect has an accelerometer built in to register the direction of gravity. Using this instead of sensing the floor plane could provide an added robustness against unleveled floors or obstructed or partial views of the floor, assuming future versions of the Kinect SDK include this functionality. The classification of moves can also be expanded to a more macroscopic level. In addition to the micro-move classifier used in this study, the algorithm could be expanded to incorporate a decision tree to recognize more complex movements. In this case, a tree would use multiple classifier steps to classify parts of the movement into their atomic actions and making classification decisions based on the combination of results. This is the visualized end goal of this type of classification.

Another future set of work to be completed before the final project is realized is the expansion of the library of moves. Most moves start with a position and/or a combination of these initial moves used in this study with slight variations, thus the current set was effective for proof-of-concept work, but not as a full library. The algorithm would become more robust with added variations of each move included in the dictionary, such as dance moves with the other foot, significant stylistic changes in the same move, etc. Further looks into structured or semi-structured decision making is an interesting and promising path to take to create a larger library. There are only so many ways to start certain moves in ballet, and the pool of possible choices to identify a dance move is drastically reduced if the first steps of that move are identified early. In addition, an end-user product could have a learning feature to add stylistic changes to the library specific to the end user, and even create new moves specific to the studio or persons using the product.

Several features could also be added to aid a dancer as a choreography tool. Assuming most dances are put to music and/or beat, the process of identifying the beat in a music and synchronizing the dance movements to that beat would aid a choreographer greatly, as the program could not only identify each move, but assign the numbered count in the music on where that move occurs. This could greatly aid dancers recording choreography, as the output of the program could almost be used without additional work. This could be done by algorithms taking input from the Kinect's microphone array, or even allowing an option to play a song from the CD tray of the computer and simultaneously running an algorithm to "sense the beat" of the song.

This work doesn't have to be limited to choreography, and the Kinect's capabilities and the ability to recognize actions in structured areas like ballet has other potential uses as well. In the realm of dance, this technology could be used in diagnostics and coaching as well as choreography, and could allow a computer to critique a dancer's form if programmed properly. Consequentially, technology like this could free time in the schedule of instructors in a dance studio or even allow a dance student to practice proper technique at home with feedback. This idea doesn't need to be limited to dance: any areas that require prescribed movements of the body could also benefit from this technology.

## VII. Conclusion

Computer vision applications are wide and diverse and include the topic of dance recognition. A potential new use of computer vision was proposed as a choreography tool. This project is a proof-of-concept, so a program that could recognize atomic ballet movements was created and explored. The Microsoft Kinect and the Kinect SDK were used to segment and track bodies and extract joint locations. These locations were transformed by rotating the axes relative to the floor, and altering the joint coordinates into link angles describing their orientations. The recognition algorithm was simple, but both major algorithms produced results above 90% using leave-one-out cross validation, and the NN algorithm was found to successfully classify test results of unknown subjects with unforeseen variations. Given these results, the project can expand in multiple areas, including further development in the recognition algorithm and the GUI as well as expansion from choreography to technique or other forms of dance. Overall, this work shows promise to expand yet again to other exciting and useful applications.

## VIII. Acknowledgements

The authors would like to thank several people for their contributions to this work. Thanks to Nick Walczak, Josh Fasching, and Duc Fehr, and others in the University of Minnesota Center for Distributed Robotics for their contributions to this work. Thanks also to Sabrina Hermsen for her shared expertise in dance and ballet and Britney Dancs for her significant contributions as a test subject during development work. The authors would also like to thank the additional dancers that volunteered to provide data for testing purposes. This material is based upon work supported by the National Science Foundation through grants #IIP-0443945, #CNS-0821474, #IIP-0934327, #CNS-1039741, and #SMA-1028076.

## References

[1] J. Aggarwal and Q. Cai, "Human motion analysis: A review," in *Non-rigid and Articulated Motion Workshop, 1997. Proceedings., IEEE*, June 1997, pp. 90 –102.

[2] L. Deng, H. Leung, N. Gu, and Y. Yang, "Recognizing dance motions with segmental svd," in *Proceedings of the 20th International Conference on Pattern Recognition (ICPR)(Istanbul, Turkey, 2010)*, 2010, pp. 1537–1540.

[3] B. Peng and G. Qian, "Binocular dance pose recognition and body orientation estimation via multilinear analysis," in *2008 CVPRW'08 IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*. IEEE, 2008, pp. 1–8.

[4] F. Guo and G. Qian, "Dance posture recognition using wide-baseline orthogonal stereo cameras," in *2006 FGR 2006 7th International Conference on Automatic Face and Gesture Recognition*. IEEE, 2006, pp. 481–486.

[5] L. Campbell and A. Bobick, "Recognition of human body motion using phase space constraints," in *1995 Proceedings., Fifth International Conference on Computer Vision*, June 1995, pp. 624 –630.

[6] S. Tsuruta, Y. Kawauchi, W. Choi, and K. Hachimura, "Real-time recognition of body motion for virtual dance collaboration system," in *17th International Conference on Artificial Reality and Telexistence*. IEEE, 2007, pp. 23–30.

[7] R. Sivalingam, G. Somasundaram, V. Bhatawadekar, V. Morellas, and N. Papanikolopoulos, "Sparse representation of point trajectories for action classification," in *Proceedings of the IEEE International Conference on Robotics and Automation*. IEEE, 2012.

[8] "Statistics: General us education and dance education," National Dance Education Organization. [Online]. Available: http://www.ndeo.org/content.aspx?page_id=22&club_id=893257&module_id=55774

[9] C. Burbank, "Dance designer - choreography tools for dance, cheer and figure skating," Choreo Technology LLC. [Online]. Available: http://www.choreopro.com/

[10] J. Shotton, A. Fitzgibbon, M. Cook, T. Sharp, M. Finocchio, R. Moore, A. Kipman, and A. Blake, "Real-time human pose recognition in parts from single depth images," in *2011 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2011, pp. 1297 –1304.

[11] R. Rubinstein, "Ksvd box," October 2009.

[12] C. Cortes and V. Vapnik, "Support-vector networks," *Machine learning*, vol. 20, no. 3, pp. 273–297, 1995.