The Applications of Computer Vision in Movement Analysis and Dance Notation

Natalia Pinpin

Campuswide Honors Collegium Thesis

Fall 2021

Professor Mary Corey, Professor John Crawford

Department of Dance

University of California, Irvine

ACKNOWLEDGEMENTS

I would like to sincerely thank the following people for their roles in my thesis and education.

To Professor Mary Corey, my thesis advisor, for being my anchor throughout the thesis and for seeing this 1.5-year long project through to its completion. Thank you for extending to me your support, guidance, and expertise.

To Professor John Crawford, my thesis advisor for the Winter 2021 quarter, for jumpstarting my research in human pose estimation and offering your unique perspective as a scholar in both computer science and dance.

To Mei-Chen Lu, my Labanotation teacher, for sharing your love of dance notation and inspiring me to develop a deeper appreciation for and interest in Labanotation.

To UROP for the 2020-2021 Fellowship and the grant that helped fund my Intermediate Labanotation course.

To my entire family for their endless support of my education. To my Dad for providing the Kinect and laptop that made this project successful and for sharing your love of computer science and innovation in the field with me. To my Mom for being my source of motivation and encouragement when I needed it.

ABSTRACT

Due to the lack of funding, little research has been performed in the intersection of dance and computer science. With the advancement of technological capabilities in artificial intelligence and computer vision, 3D human pose estimation is becoming an increasingly feasible and accurate mode of analyzing movement. This thesis examines the applications of 3D pose estimation through the lens of dance-like movement and Labanotation. Using the Microsoft Kinect to perform 3D pose estimation, I wrote a computer program that generates a Labanotation score from a video of someone dancing a simple phrase of movement. The program utilizes the 3D coordinates predicted by the Kinect that represent the location of human joints, and it analyzes the relationships between the arm and the leg joints to write Labanotation symbols. The final Labanotation score is then written to a PDF. By testing the program on nine distinct combinations of movement, I discovered that the program can in fact perform elementary Labanotation analysis at a reasonably accurate level. However, recurring challenges emerged, such as with temporal inaccuracies, self-occlusions of body parts, and locomotion, demonstrating that much more progress in the field of 3D pose estimation remains to be achieved. This paper highlights the potentials in using technology as a means to innovate in the dance community and in using movement and dance research as a way to inform the learning of computers.

TABLE OF CONTENTS

INTRODUCTION

As a double major in Dance and Computer Science, I have always aspired to unite these two distinct fields. Accordingly, this thesis is an interdisciplinary case study on the potential behind movement analysis and technology. In the pursuit of a project that synthesizes my two areas of study, I designed a computer program that can analyze a video of a simple phrase of movement and generate a basic Labanotation[1] score for the movement. To illustrate, the input of this program is a video of a solo dancer performing a simple movement such as a *port de bras*, that moves their arms in a path above their head, then open to the sides, then down beside their legs. The program then analyzes this clip and generates the output of a PDF file with the written notation for the dance.

By researching and designing a program that involves both dance and technology, my interest is to explore the possibilities of intersection between the two disciplines, especially in tools for digital movement analysis. Because research in dance and the arts in general is highly underfunded, this area of study between dance and artificial intelligence is fledgling yet small. Regardless, I have learned that there are methodical approaches to researching movement, and this can have implications beyond the field of dance, extending to healthcare and physical therapy or surveillance and security.

[1] Labanotation: a system for notating movement; Section III goes into detail about the basics of Labanotation.

CHAPTER 1

I. Literature Review

In the past couple of decades, researchers internationally have explored the intersection between Labanotation and computer science, looking for innovations to support Labanotation as a tool for teaching and to facilitate the composition of scores. Beginning in 2003, Lars Wilke of Credo Interactive Inc., Tom Calvert of Simon Fraser University, Rhonda Ryman of University of Waterloo, and Ilene Fox of the Dance Notation Bureau collaborated on LabanDancer, a computer application that translates a Labanotation score into a visual, animated representation of the choreography. LabanDancer interprets scores created in LabanWriter, an existing computer application developed at The Ohio State University to write and edit Labanotation scores.[2] Taking an existing score, LabanDancer parses through the symbols, translating them into gestures, support changes, and a floor plan in order to recreate the score visually (Wilke et al., 2003). When the research endeavor ended in 2005, many challenges persisted in LabanDancer, especially related to interpreting "implied" movements. Whereas a human reader is able to interpolate the anatomically logical limb orientations and transitions between movements, the computer relies on a rigorous model of human motion in order to produce plausible positions (Wilke et al., 2005).

More recently, various research groups have attempted to use computer vision to help automate the writing of Labanotation scores. Developed by Worawat Choensawat of Bangkok University, Minako Nakamura of Ochanomizu University, and Kozaburo Hachimura of Ritsumeikan University from 2014 to 2015, GenLaban and LabanEditor are counterpart programs for generating Labanotation, digitally writing the score, and viewing its animation.

[2] LabanWriter continued to be a popular application in the U.S. for writing Labanotation up to recent years, but now it is no longer supported by current operating systems and thus is becoming obsolete.

Using BVH motion capture data, GenLaban selects keyframes to analyze the movements and creates Labanotation data files readable by LabanEditor. GenLaban was tested with some success on Japanese Noh dance and on movements such as support changes and jumping. However, at the termination of this research, significant issues still remained, such as the reliance on user-selected keyframes which generated different scores based on which frames were chosen (Choensawat et al., 2014; Choensawat et al., 2015).

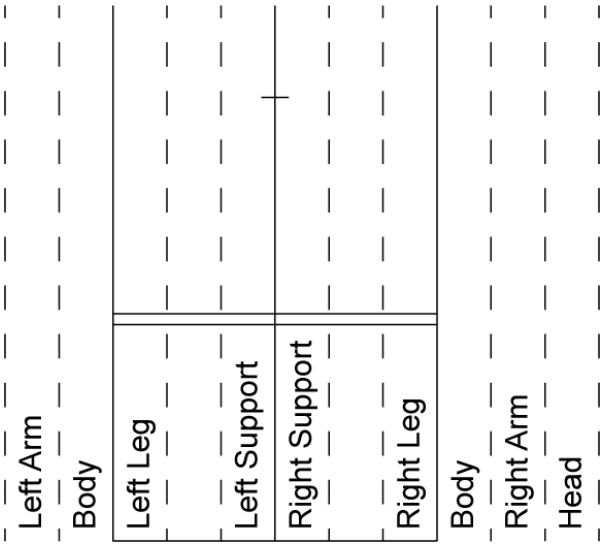
The most significant and long-term research on computer-generated Labanotation has been performed by collaborators from Beijing Jiaotong University from 2014 to the present. The work of this group focuses on producing Labanotation scores for the purpose of recording and preserving Chinese folk dances. Like the work of Choensawat et al. (2015), the process for the majority of their work begins with capturing BVH motion capture data that is then segmented and formatted into Labanotation data to construct a score (Guo et al., 2014; Zhou et al., 2016; Wang et al., 2018). Throughout the course of their research, they have incrementally improved on the accuracy and complexity of their methods of motion analysis, most recently landing on a convolutional recurrent neural network that performs both spatial and temporal motion analysis without the need for pre-segmenting individual keyframes of movement (Xie et al., 2020).

II. Background

Labanotation

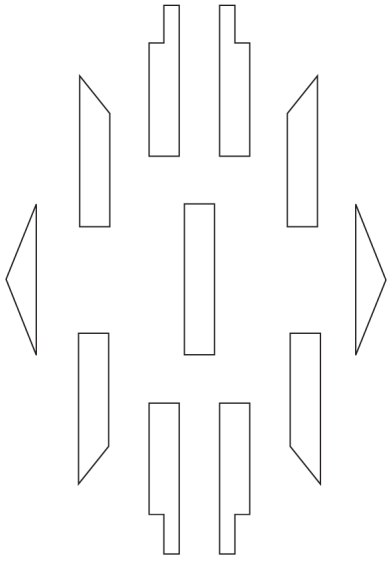
In the 1920s, Austro-Hungarian dance educator and researcher Rudolf Laban began devising the beginnings of a notation for dance, a system of written symbols that represent movement. Throughout the rest of the century, dance researchers internationally continued to develop this system of dance notation, which became known as Labanotation. In New York, the

Figure 1



Note. The labeled columns of the Labanotation staff. The dotted lines do not appear on an actual staff.

Figure 2



Note. The symbols encompassing the eight directions in Labanotation and place.

Figure 3

Note. The three levels in Labanotation: high, middle, and low respectively.

Dance Notation Bureau, established in 1940, became the hub of American Labanotation (Topaz, 1996). Labanotation is a critical system not only for the preservation of choreography but also for dance education and research. As a universal, non-verbal, and style-independent analysis of movement, it is extensible to any movement practice (Guest, 2005).

In Labanotation, symbols and their placement on a vertical staff indicate four elements: the direction, level, and timing of the movement and the body part doing the movement. Each column on the staff represents a different part of the body. The two middlemost columns describe the supports, or the body parts that support the weight of the rest of the body. Other columns depict leg and arm gestures, or movements that do not carry weight; body movements, including that of the torso and pelvis; and head movements (Figure 1).

The shape of the symbol specifies one of the eight main directions (forward, forward diagonal right, forward diagonal left, side right, side left, backward diagonal right, backward diagonal left, and backward) or place (Figure 2) while the shading of the symbol indicates one of the three levels (Figure 3). Symbols are placed into the columns to indicate a movement of that body part. In addition to these basic symbols, the Labanotation vocabulary includes a wide variety of other symbols and signs to indicate tilting, turning, bending of limbs, and other combinations of movement. Time progresses upwards on the staff with the relative length of the symbol indicating the timing. A complete score also contains a floor plan that illustrates how the movement travels around the entire stage area (Guest, 2005).

Human Pose Estimation

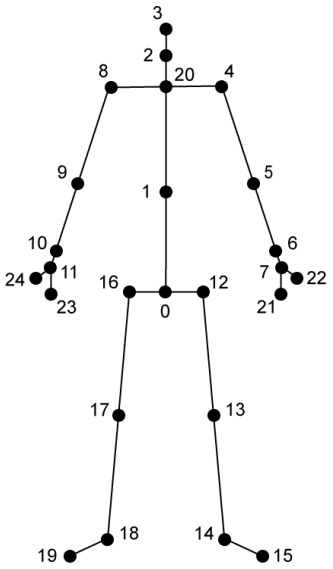
Machine learning is a branch of artificial intelligence where the computer tries to learn a certain behavior or pattern, improving itself as it acquires more information. This process of

learning from large amounts of data is called training a model. The computer must use complex statistical algorithms to classify and arrive at conclusions about data it has not seen previously; this process of producing output is known as prediction. Computer vision is a branch of machine learning that involves the interpretation of image and video data; essentially, the computer learns how to "see" and analyze images like a human (Zhang et al., 2016).

In particular, I worked with human pose estimation, the identification of the position of human body parts in an image. Many variations of human pose estimation exist, such as 2D or 3D estimation, single or multi-person estimation, and monocular (one camera) or multi-camera input (Liu et al., 2015). For every version of this human pose estimation problem, the basic goal is to extract information, or features, about the position of the people from one or more images. For video input, the video must first be separated into keyframes or individual snapshots of images that will be used as the data. For each image, the first step is the identification of the people in the image, often done by segmentation. In segmentation, a contour is drawn around the silhouette of each person in order to determine their exact position in the image. At the end of the process, refined high-level features that contain information about the individual limbs and joints of the body are found. These joints, or keypoints, of the body are modeled by the structure of the data. A common method of representing joints is a kinematic or skeletal model that depicts the body as a graph of connected joints (Figure 4) (Gong et al., 2016).

One of the most popular methods of human pose estimation is deep learning, a type of machine learning that uses many layers of processing data into features. Deep learning is implemented by artificial neural networks (ANN) or more simply known as neural networks, a mathematical model of learning that replicates the network of neurons in a brain. ANNs simulate neurons with mathematical functions that transform the data as it travels from layer to layer in

Figure 4



0 Spine base
1 Spine mid
2 Neck
3 Head
4 Shoulder left
5 Elbow left
6 Wrist left
7 Hand left
8 Shoulder right
9 Elbow right
10 Wrist right
11 Hand right
12 Hip left
13 Knee left
14 Ankle left
15 Foot left
16 Hip right
17 Knee right
18 Ankle right
19 Foot right
20 Spine shoulder
21 Hand tip left
22 Thumb left
23 Hand tip right
24 Thumb right

Note. The 25-joint skeletal model utilized by the Microsoft Kinect v2. Each joint is mapped to a 3D coordinate that represents where the joint is in space, relative to the Kinect camera.

the network. There are many different types of ANNs such as recurrent neural networks, deep neural networks, and convolutional neural networks (Krenker et al., 2011). In order to improve the accuracy of human pose estimation, ANNs train on large datasets of annotated images of humans. The most well-known, open-source 2D human pose estimation datasets are FLIC, LSP, MPII Human Pose, and COCO. Containing between 5,000 to 200,000 images, these datasets are not only used to train the networks but also to assess their performance (Munea et al., 2020).

CHAPTER 2

III. The Program

Intersecting these two distinct areas of pose estimation and Labanotation, my research culminated in the creation of a program that generates Labanotation from a video of someone performing a phrase of movement. Because Labanotation is a concrete method of representing an abstract concept like movement, it acts as a bridge between the seemingly whimsical nature of dance and the rigid, precise parameters necessary to artificial intelligence. Furthermore, the way Labanotation organizes and analyzes the body is analogous to body segmentation in computer vision, making it a compatible lens to develop my research around. Throughout this coding process, aspects of the two branches of study informed each other; characteristics of Labanotation influenced the logical structure of the code while the pipeline of transforming the video to a score determined how Labanotation information was created and processed. The limitations of the pose estimation also narrowed the scope of the Labanotation, shaping the focus of the research and the examination of Labanotation.

In order to begin this project, I first needed to familiarize myself with reading and writing Labanotation. Through Zoom classes offered by the Dance Notation Bureau, I completed their Elementary and Intermediate Labanotation courses. At the Elementary level, I covered the fundamentals and topics including foot supports, basic arm and leg gestures, aerial work, turns, pathways, and floor plans. At the Intermediate level, I learned how to notate more detailed movement–such as flexion, extension, turning, and tilting–of specific body parts including the individual limbs, the torso and pelvis, and the feet, as well as how to notate sitting, kneeling, and lying down. While my computer-generated Labanotation does not encompass concepts beyond that of the Elementary course, a more comprehensive understanding of Labanotation provided

me the practice and insight necessary to design code that is compatible with the organization of Labanotation and how it breaks down movement.

At the beginning of Winter quarter 2021, I began the coding component of my research. With a list of pre-trained pose estimation models compiled by Zhe Wang[3] as a starting point, I began by surveying the current work in 3D pose estimation. I ultimately decided to use the Microsoft Kinect to execute the 3D pose estimation. In the following sections of this paper, I will discuss the Kinect in greater detail.

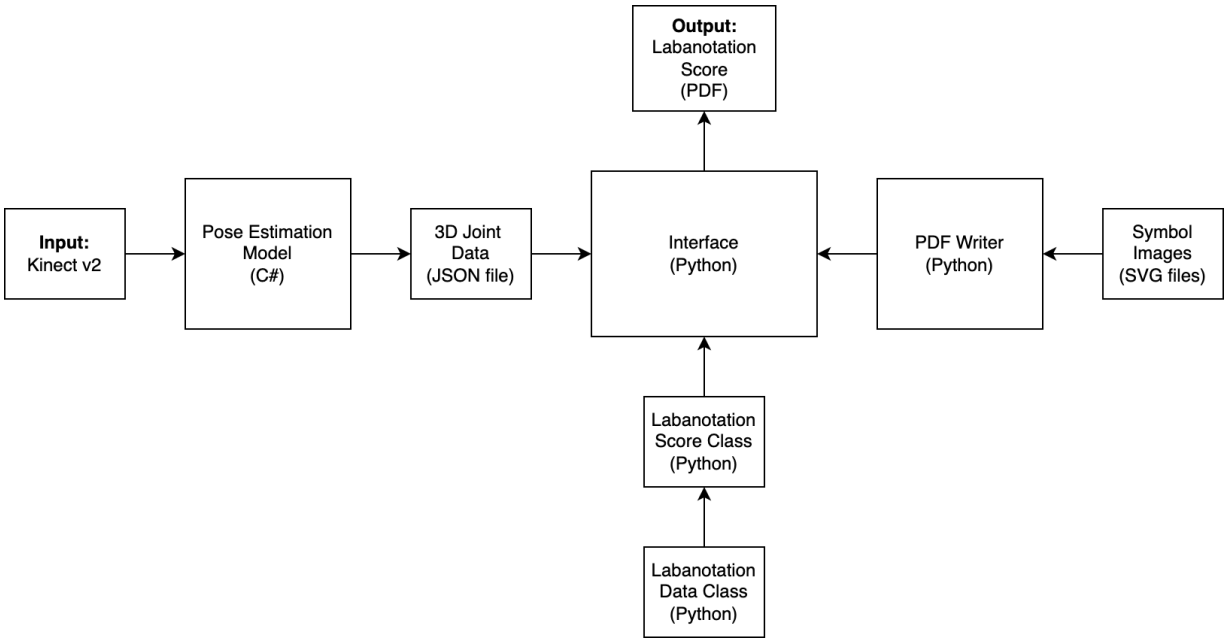
The Code

My code is broken down into separate modules: the pose estimation model, the interface with the pose estimation model, the Labanotation data and score classes, and the Labanotation score PDF writer (Figure 5).[4] In this code, the process of translating a video into a Labanotation score is the following: first, a pose estimation model uses the frames from a video of a person performing a phrase of movement to produce a sequence of 3D body coordinates. These coordinates are then turned into Labanotation data objects that correspond to the symbols on the Labanotation staff. The resulting Labanotation data objects are assessed and organized in a Labanotation score object. After the completion of the analysis of each frame of movement, the program interprets the Labanotation score object, writing images onto a PDF to compose the score.

In order to streamline this video-to-Labanotation process, I set strict requirements for the video input. The video must have exactly one person fully in frame, performing the movement to be analyzed. The tempo of the movement is performed at 60 beats per measure, meaning each

[3] https://github.com/wangzheallen/awesome-human-pose-estimation
[4] The full code can be accessed at https://github.com/nataliapinpin/PoseEstimationLabanotation

Figure 5



Note. Flowchart of the modules and intermediate files used in the program I wrote.

beat in the phrase lasts for one second. The phrase of movement must run at most eight measures long, and the dancer must face directly forward towards the camera. As for the choreography, the dancer's movements must stay within the limits of the highly elementary Labanotation abilities of the program. The program can only perform gestural analysis of the arms and legs and cannot analyze contraction of the limbs. In other words, the dancer's motions are limited to arcing pathways of completely straight, extended arms and legs. Another result of these limitations is that the dancer must stay in the same location for the duration of the video; they cannot move around the room.

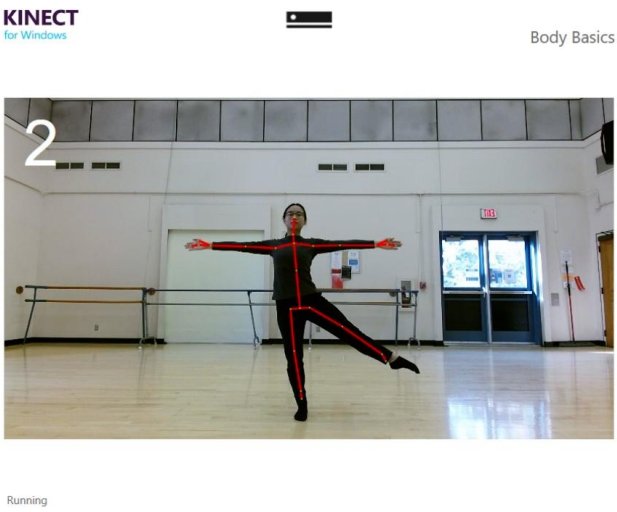
Throughout the course of this thesis, I explored various 3D pose estimation models; two in particular were integral in the development of this program. In the first half of the project, I

utilized a model developed by Zhou et al. (2017)[5] because of its compatibility, efficiency, and ease of incorporating it into my own code. The creation of input data began with recording videos of myself performing a phrase of movement and editing the clips with an external video editing software. To generate images compatible with the pose estimation model, I wrote code that parsed the edited video input into keyframes. These keyframes were taken every second of the video and saved into a directory on the computer. To perform the pose estimation, I repurposed the demo.py code provided in the original code; by providing an image to the model, it returns a list of 3D coordinates for the joints of the body. While easy to work with in code, this model proved to be highly inaccurate and insufficient for the pose estimation necessary for Labanotation.

When I was part-way through this thesis, I obtained access to a Kinect for Xbox One (Kinect v2) and a compatible Windows PC. The Microsoft Kinect sensors have depth detecting cameras that were primarily created to aid in immersive, motion-controlled Xbox games but have been popular with developers because of their (at the time) advanced motion-sensing technology (Zhang, 2012). The history and technology of the Kinect sensor is further discussed in Section V. One significant benefit of the Kinect sensor is that it performs pose estimation in real-time, meaning the video data gathered from the Kinect is immediately processed and converted into 3D coordinates without the need for me to manually go through intermediate steps to prepare the video data. I built my code into a sample project from the Kinect SDK 2.0,[6] that handles connecting the Kinect to the PC, gathering the necessary video streams from the Kinect camera, and performing the pose estimation. The Kinect SDK 2.0 comes with sample code in the

[5] https://github.com/xingyizhou/Pytorch-pose-hg-3d
[6] SDK (software development kit): a set of software tools built to assist developers; I used code from the Kinect SDK 2.0: https://www.microsoft.com/en-us/download/details.aspx?id=44561

Figure 6



Note. A screenshot of the pop-up window that visualizes the 3D pose estimation in real-time. The image is mirrored so that the person performing the movement can watch themselves on screen and ensure their movements align with the count at the top left corner.

languages C# and C++ for various features on the Kinect, including working with audio, color frames, and depth frames. As a starting point, I used the BodyBasics-WPF sample project, which handles 3D pose estimation and visualization of the results. The visualization occurs in a pop-up window where the user can see the real-time skeletal model of the estimated joints. I modified the existing code to overlay the estimated joints on top of the frame captured by the RGB camera (Figure 6).

Since the Kinect produces real-time pose estimation, the dancer whose movements are analyzed performs a phrase of movement, and the Kinect instantaneously outputs the estimated 3D coordinates of the joints. When the dancer completes their phrase, they simply close the pop-up screen to indicate that they have finished, and the program saves the 3D coordinates into a JSON (JavaScript Object Notation) file before ending. A JSON file is a type of formatted text

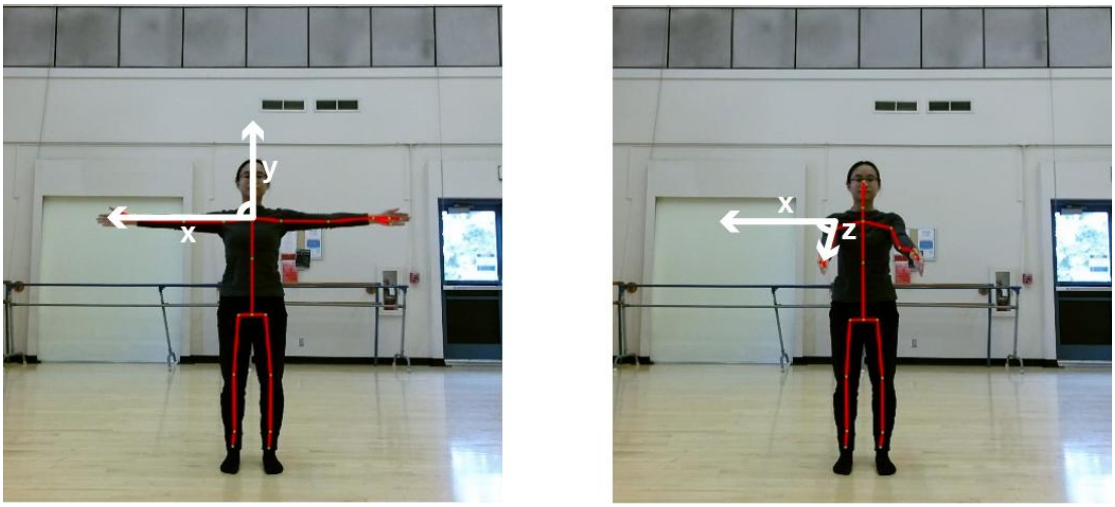
file. It is highly compatible with this data because its format can map each body joint to a 3D coordinate, and this structure of data is easily preserved even when read into the Python program that generates the Labanotation. Finally, to aid the dancer in the real-time performance of movement, I incorporated a timer that maintains the mandatory 60 bpm tempo. When the program begins, a countdown of eight beats appears, signaling when the dancer should begin the choreography. Throughout the phrase, an ongoing count from beat one to eight is displayed in the top left hand corner of the pop-up window so that the dancer stays in tempo (Figure 6).

For the next module of my program, I constructed a LabanotationData class that represents the information for the symbols of the Labanotation score. Given a column, starting beat, and joint coordinates, this class can calculate the corresponding Labanotation symbol, namely the direction and level of the movement. This organization of this class was inspired by the Labanotation data structure devised by the Labanotation research at Beijing Jiaotong University (Guo et al., 2014). The program computes the direction and level of a limb using the angular distance between the limb's fixed and free joints and either the x-axis or y-axis. More specifically, it measures the angle created between an arm or leg and the x-axis for the direction, and it measures the angle created between an arm or leg and the y-axis for the level (Figure 7).

Using the LabanotationData class, I then constructed a LabanotationScore class that organizes the individual pieces of data into the columns of the staff. In addition to storing the symbol data, it analyzes the symbols to determine the more complex structure of the staff, especially in the support column where hold symbols systematically appear in lieu of the basic Labanotation symbols.

To complete the pipeline from pose estimation to Labanotation, the next module writes the Labanotation score into a PDF. This module takes the LabanotationScore object and reads the

Figure 7

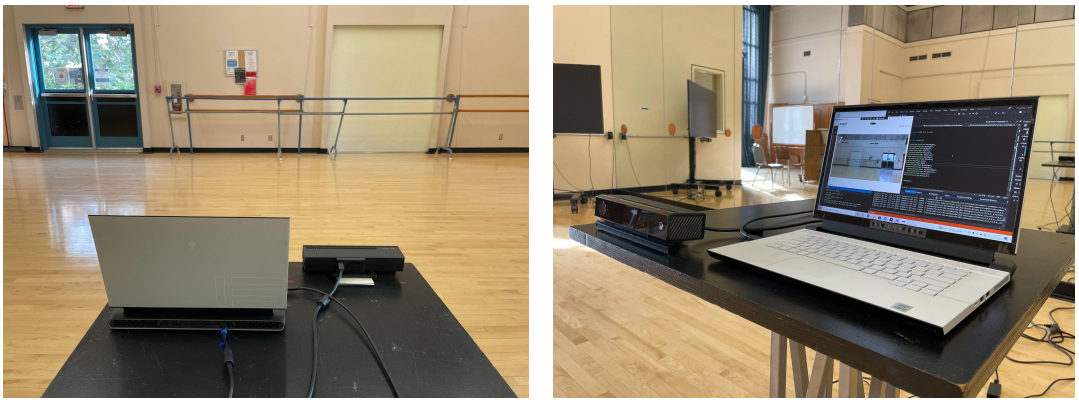


Note. The left image depicts the angular distance in the xy-plane calculated for the level of the right arm gesture. The right image depicts the angular distance in the xz-plane calculated for the direction of the right arm gesture.

information about the symbols and their positions on the staff. After determining this symbol and position, it writes a SVG (scalable vector graphic) file of the Labanotation symbol onto a PDF. SVG and PDF are vector-based file formats that better support line-based drawings, like Labanotation symbols, as opposed to JPG and PNG files that are bitmap-based and more compatible with photos. To produce SVG files for each basic Labanotation symbol, I utilized the Adobe Illustrator Labanotation templates available from the Centre national d'écriture du mouvement en cinétographie Laban (CNEM) website, the center of Kinetography Laban in France.[7] Finally, the program determines the number of measures in the score, adding a double barline at the end of the staff, and numbers the measures, writes the time signature, and labels the starting facing. With the finished output, it saves a PDF of the score onto the computer.

[7] https://www.cnem-laban.org/publications?fbclid=IwAR0GAQoBPSwgFVcIBuNvTUlP93nUwIK2fi2Z19u4du7YVQEaT2Z5KzSL0h8

Figure 8



Note. The setup for the final recording of test combinations.

The last component of the program is the pose estimation interface. In order to process the JSON file of 3D coordinates into a Labanotation score, this code acts at the interface that connects the output of the pose estimation model to the two sections of code I just discussed that process data into Labanotation. I run this code to execute the entire section of the program that analyzes the 3D coordinates and turns them into Labanotation symbols.

IV. The Output

To test the program, I wrote nine 4-measure Labanotation scores: three arm combinations, three leg combinations, and three combinations using both the arms and legs. Each combination was choreographed to target a particular pattern of movement. Below, where I present the results of each combination, I discuss my intention when designing that combination. For the final filming, I performed each combination three times, capturing the data with the Kinect (Figure 8).

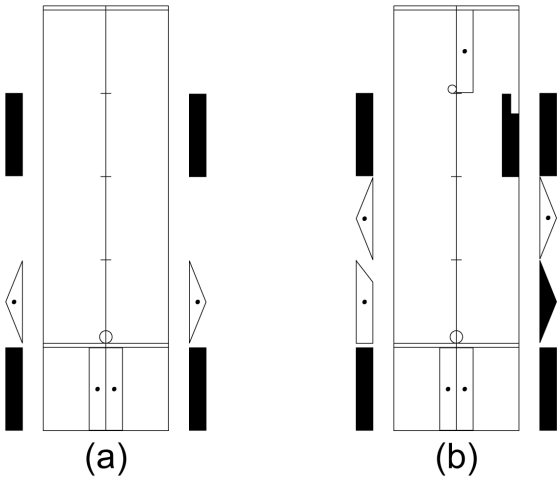
To assess these results, I devised a quantitative method of grading. The right arm, left arm, right leg, and left leg are each assigned a grade out of 34. For combinations 1-3, I only

graded the arm gestures, and for combinations 4-6, I only graded the leg gestures and supports, making a total grade out of 68 for each computer-generated Labanotation score. For combinations 7-9, I assessed both the arms and legs, making a total grade out of 136 for each score.

The 34 points per arm or leg are broken down into 2 points per beat, since each Labanotation score is 17 beats long (one beat for the starting position plus four beats per four measures). For each beat, I look at the direction and level, giving one point for each correctly predicted. For example, if the right arm at beat one is predicted to be at side low and the actual Labanotation score shows the right arm at side middle, the right arm would receive one point for the direction but zero points for the level, totalling one out of the two possible points for that beat. For the legs, if the symbol is in the incorrect column (i.e., in the support column instead of the gesture column or vice versa), the beat automatically receives zero points. See Figure 9 for a more detailed demonstration of the grading procedure. After grading all three computer-generated Labanotation scores, I took the average of the three to create an overall accuracy percentage for each combination.

Below are the results for each of the combinations. I will refer to the correct score that I manually wrote as the “ground truth” score, and I will refer to each of the computer-generated scores as the “prediction” scores. Each combination has one ground truth score and three prediction scores that correspond to the three separate times I filmed myself performing the combination. If a prediction score is exactly identical to the ground truth score, meaning it contains zero errors and received full points, I write “Same as Ground Truth” instead of reproducing the entire ground truth notation. I also indicate and provide images of notable errors in some of the prediction scores, which are elaborated on more in the discussion section.

Figure 9



Note. An example to demonstrate grading on one measure of a Labanotation score. Assume (a) is the ground truth and (b) is the prediction that we are grading. There are five beats in total to grade, resulting in 40 possible points.

0. The starting position is identical for both (a) and (b), so beat 0 gets all 8 points.
1. In beat 1, the left arm has the correct level but incorrect direction, getting only 1 out of 2 points. The right arm has the correct direction but incorrect level, getting only 1 out of 2 points. The leg supports are correct. Thus, beat 1 gets 6 out of 8 points.
2. In beat 2, both arms adjust to the correct position and the supports remain the same, so beat 2 gets all 8 points.
3. In beat 3, the right leg gestures to forward low, causing it to automatically receive zero out of 2 points because it is supposed to be a support. The rest of the gestures are correct, resulting in 6 out of 8 points for beat 3.
4. In beat 4, the right leg returns to the correct support position, so beat 4 gets all 8 points.

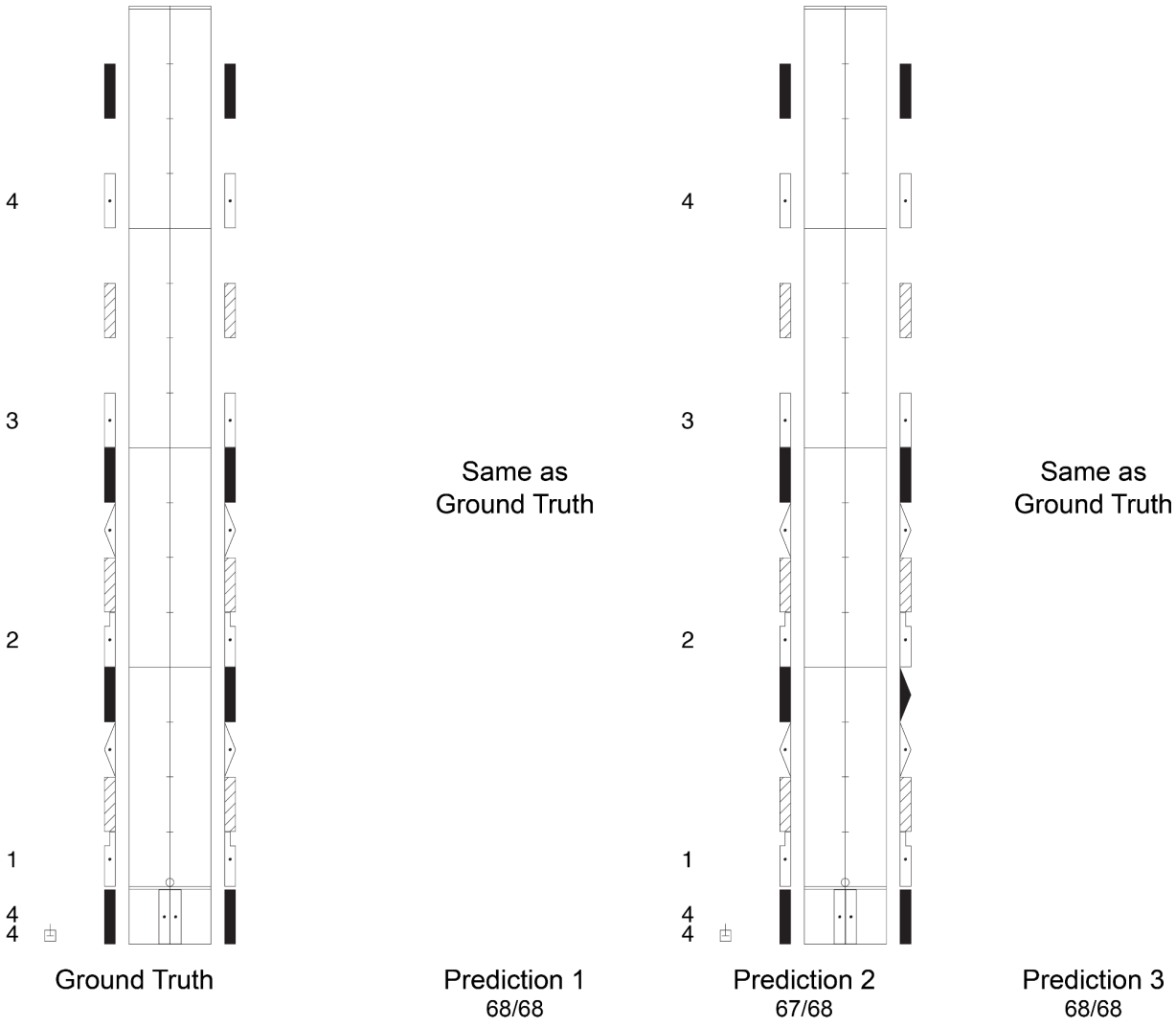
The last step is to sum up the points: 8 + 6 + 8 + 6 + 8 = 36 / 40 points.

Each of the following results pages contains a link to a video of me performing the eight-measure phrase of movement. For a playlist of all nine of those combination videos, visit https://www.youtube.com/playlist?list=PLe3-971G5fBqKMuS6gONNZHUnpLRZZKP3. Alternatively, if you are not viewing this thesis in a medium that allows you to use links, scan the below QR code to access the playlist. Note that the videos and images of the combinations in this thesis are mirrored (i.e., my right arm is located at the right side of the image).

Combination 1: Arms 1

The first two measures of this combination test a continuous, circular motion of the arms. The last two measures test place high, middle, and low.

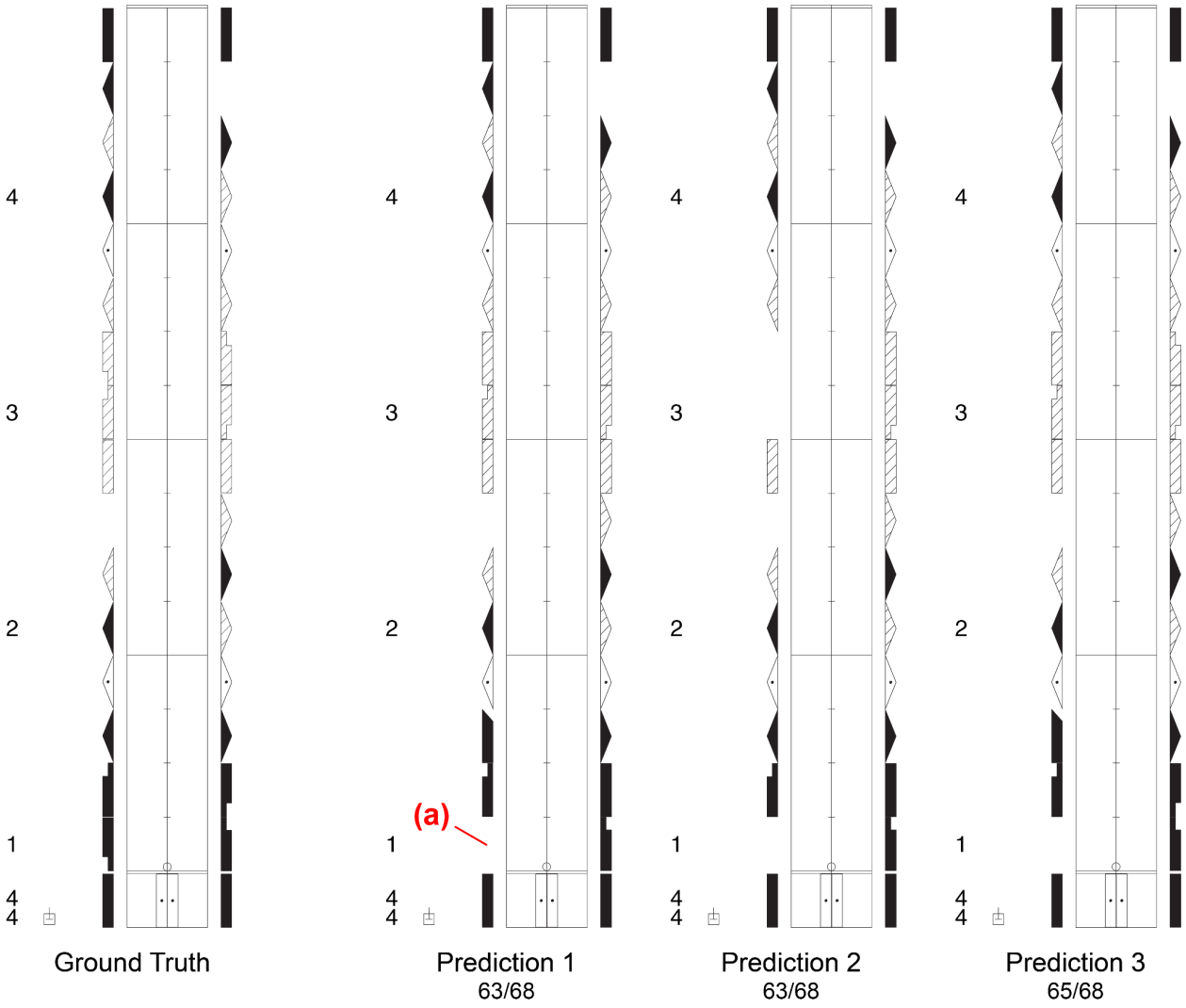
Overall Accuracy: 99.5%



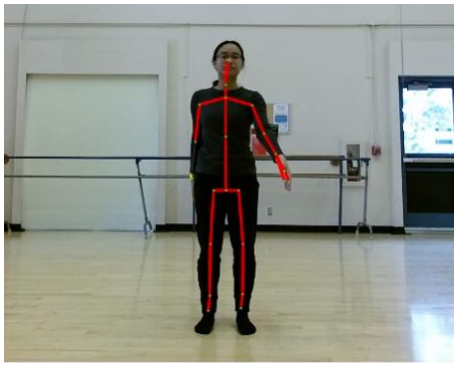
Combination 2: Arms 2

This combination targets low and high level positions of the arms in various directions.

Overall Accuracy: 93.6%



(a)



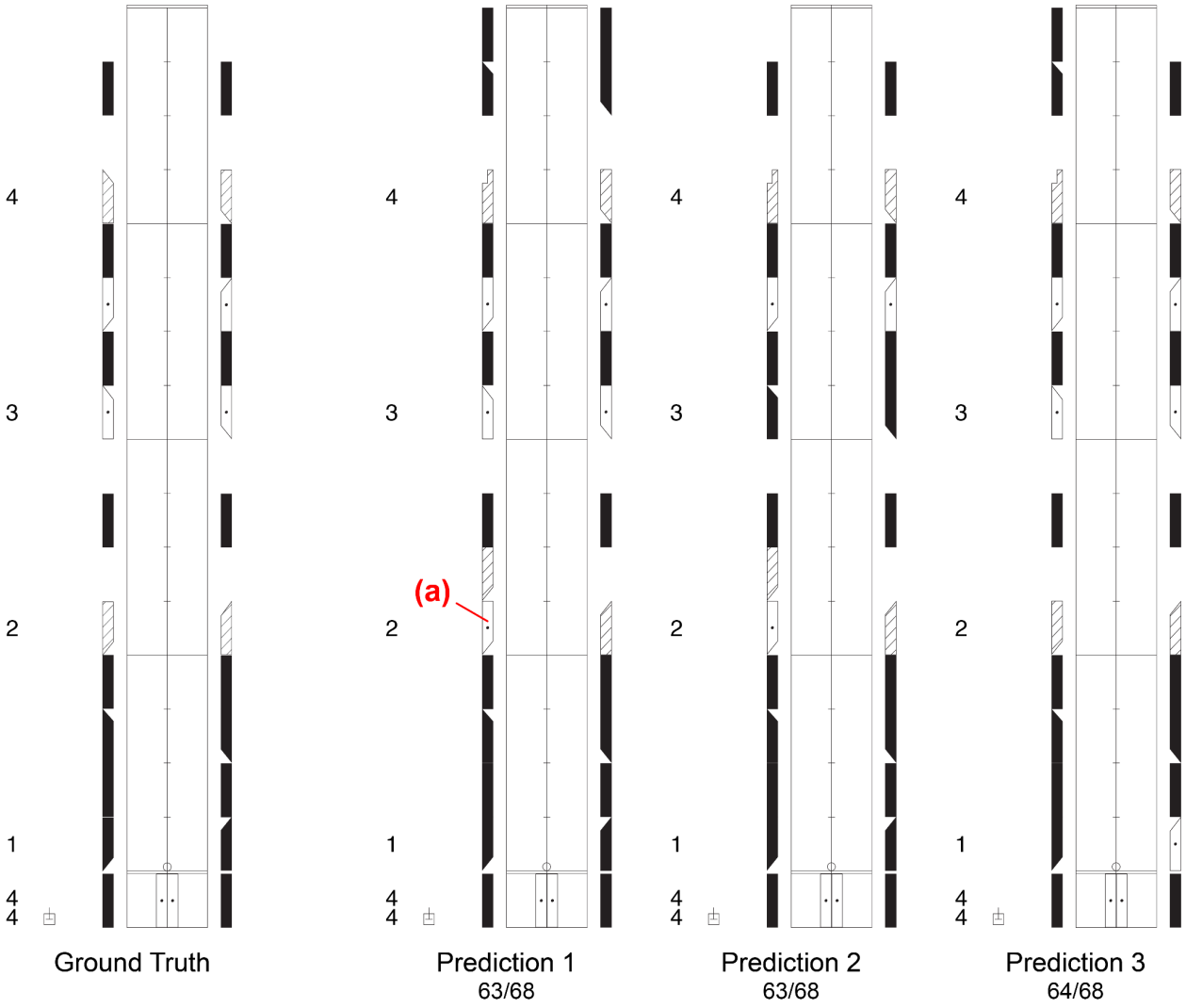
Note. Self-occlusion of the left arm as it moves to backward low causes movement to be incorrectly identified as place low.

This image is mirrored (i.e., my left arm is located at the left side of the image).

Combination 3: Arms 3

This combination assesses the diagonal directions of the arms at various levels.

Overall Accuracy: 93.1%



(a)

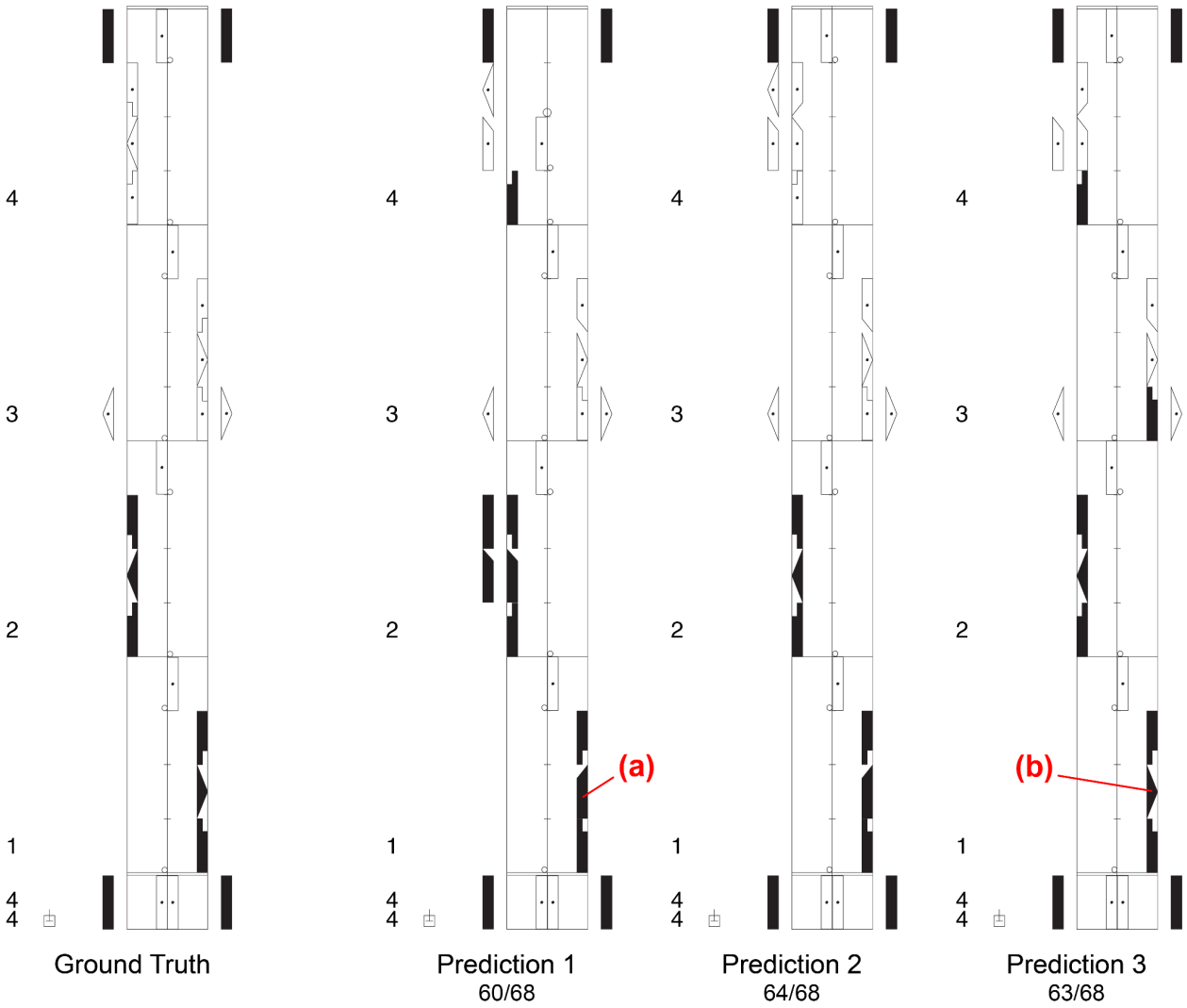
Note. The left arm is late to reach backward diagonal left high. This imprecision of movement causes beat five to incorrectly be at middle level and beat six to then correctly resolve to high level.

This image is mirrored (i.e., my left arm is located at the left side of the image).

Combination 4: Legs 1

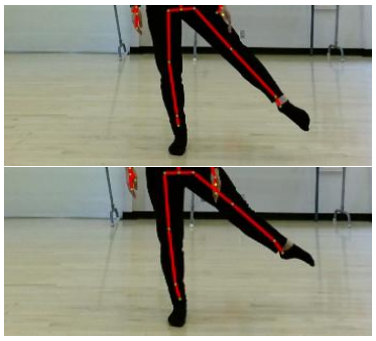
This combination tests a continuous, circular motion of the legs, known in ballet as a *rond de jambe*. This is first performed at the low level on each leg and then at the middle level.

Overall Accuracy: 91.7%



(a)

(b)



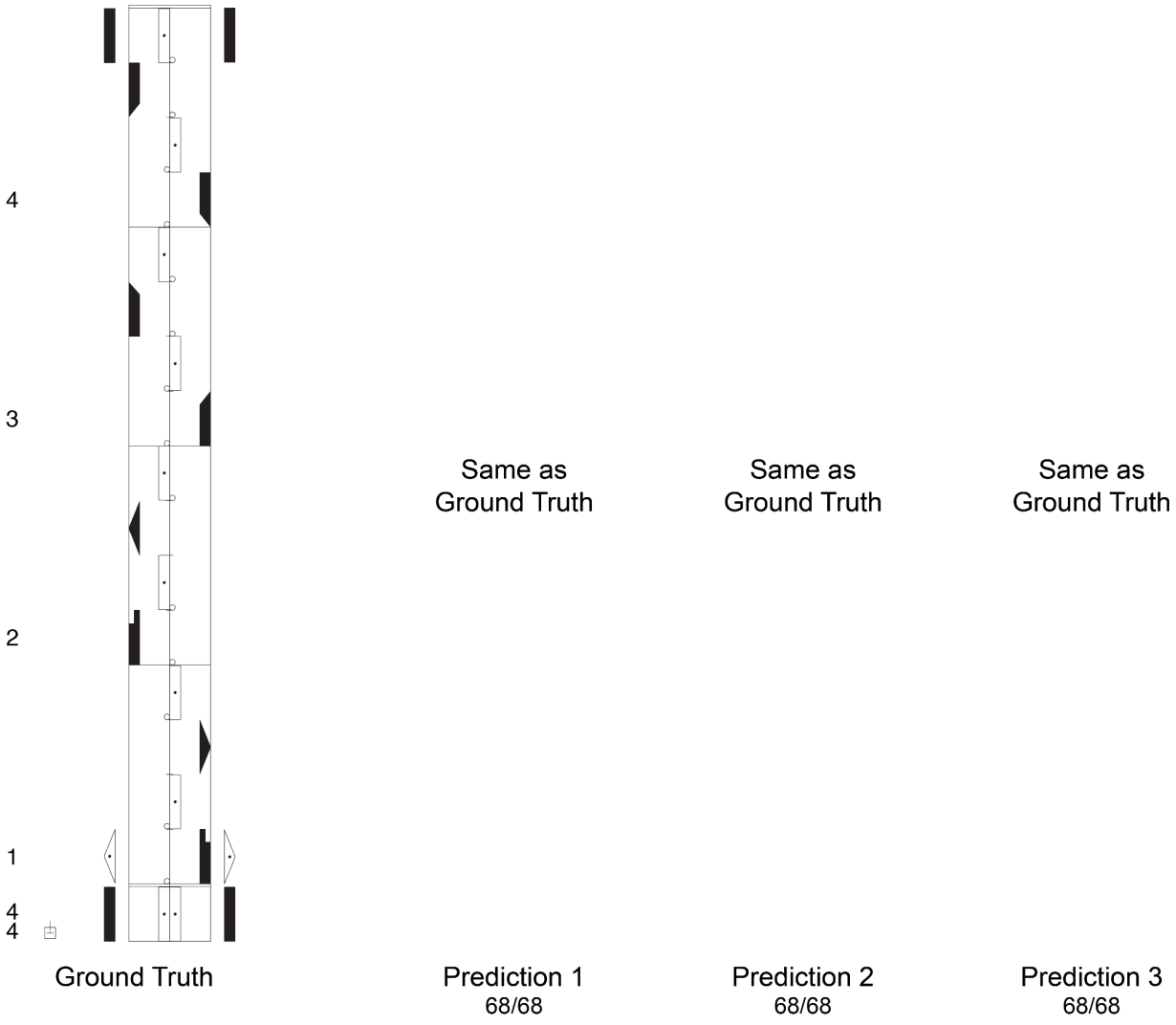
Note. In (a), the right leg is incorrectly predicted to be at right forward diagonal low whereas in (b), the right leg is correctly predicted to be at right side low. Slight variations in the performance cause these different results.

These images are mirrored (i.e., my right leg is located at the right side of the image).

Combination 5: Legs 2

This combination features alternating leg motions in various directions that brush slightly off the floor, known in ballet as a *dégagé*. Between each movement, the leg closes into place middle in the support column.

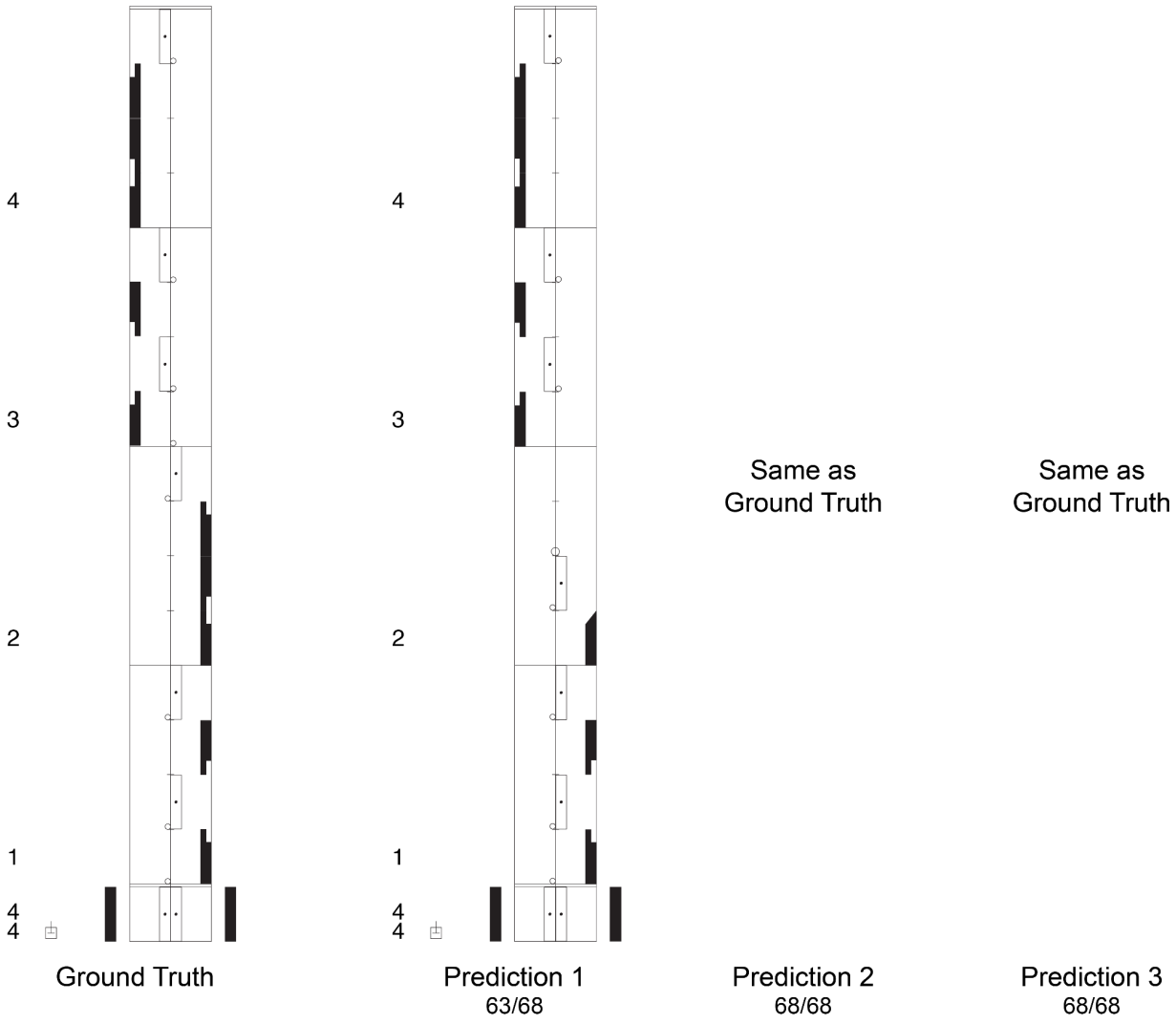
Overall Accuracy: 100%



Combination 6: Legs 3

This combination once again utilizes the *dégagé* but experiments with consecutive forward and backward gestures that only briefly pass through place low.

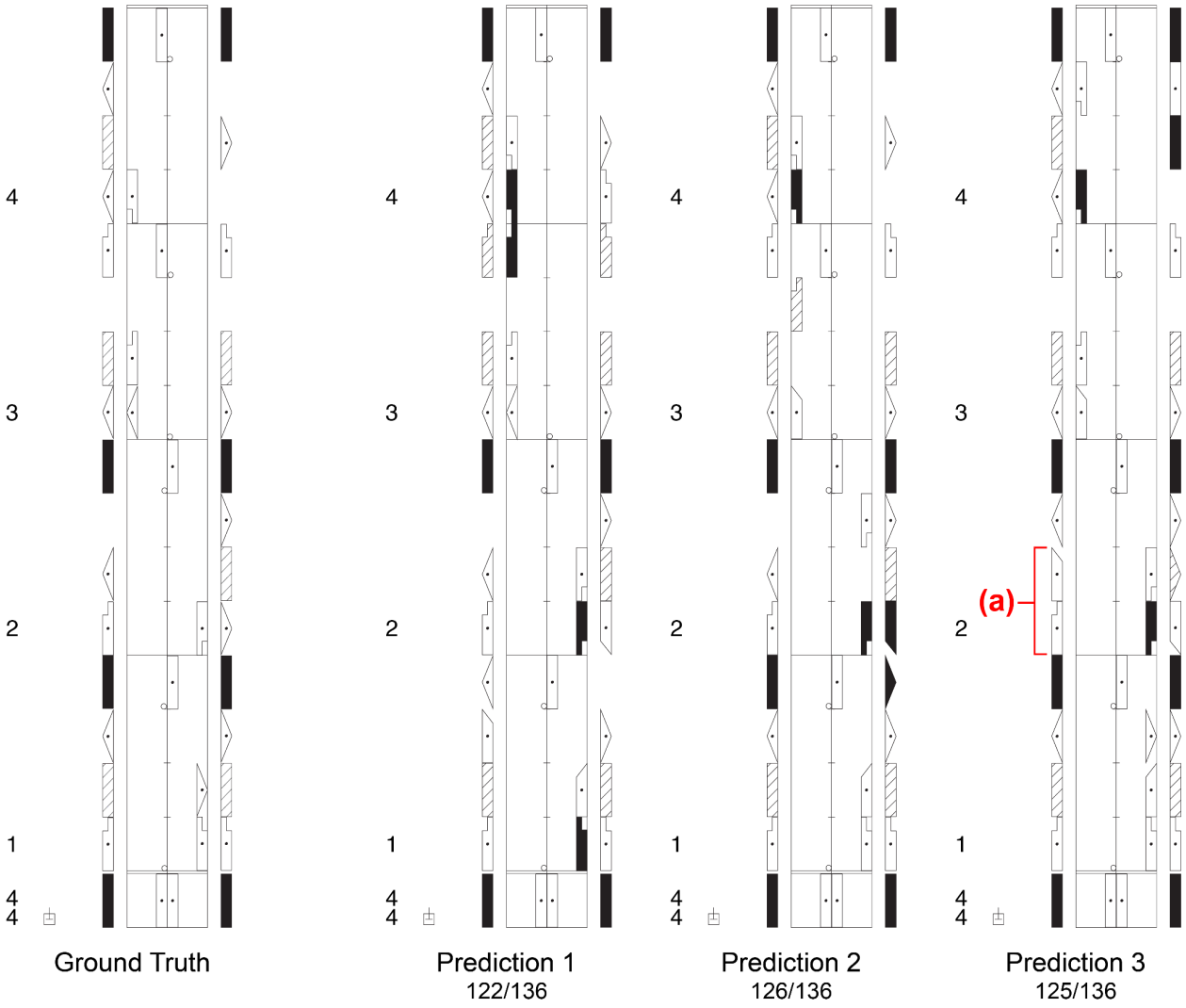
Overall Accuracy: 97.5%



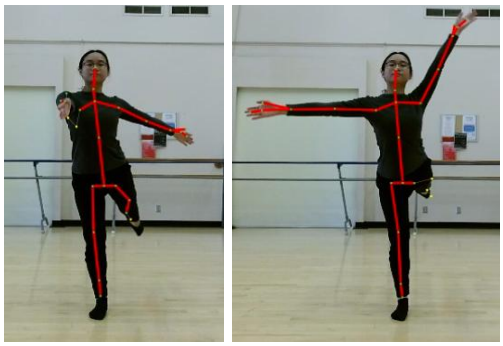
Combination 7: Arms and Legs 1

This combination reinvestigates circular motions of both the arms and legs.

Overall Accuracy: 91.4%



(a)



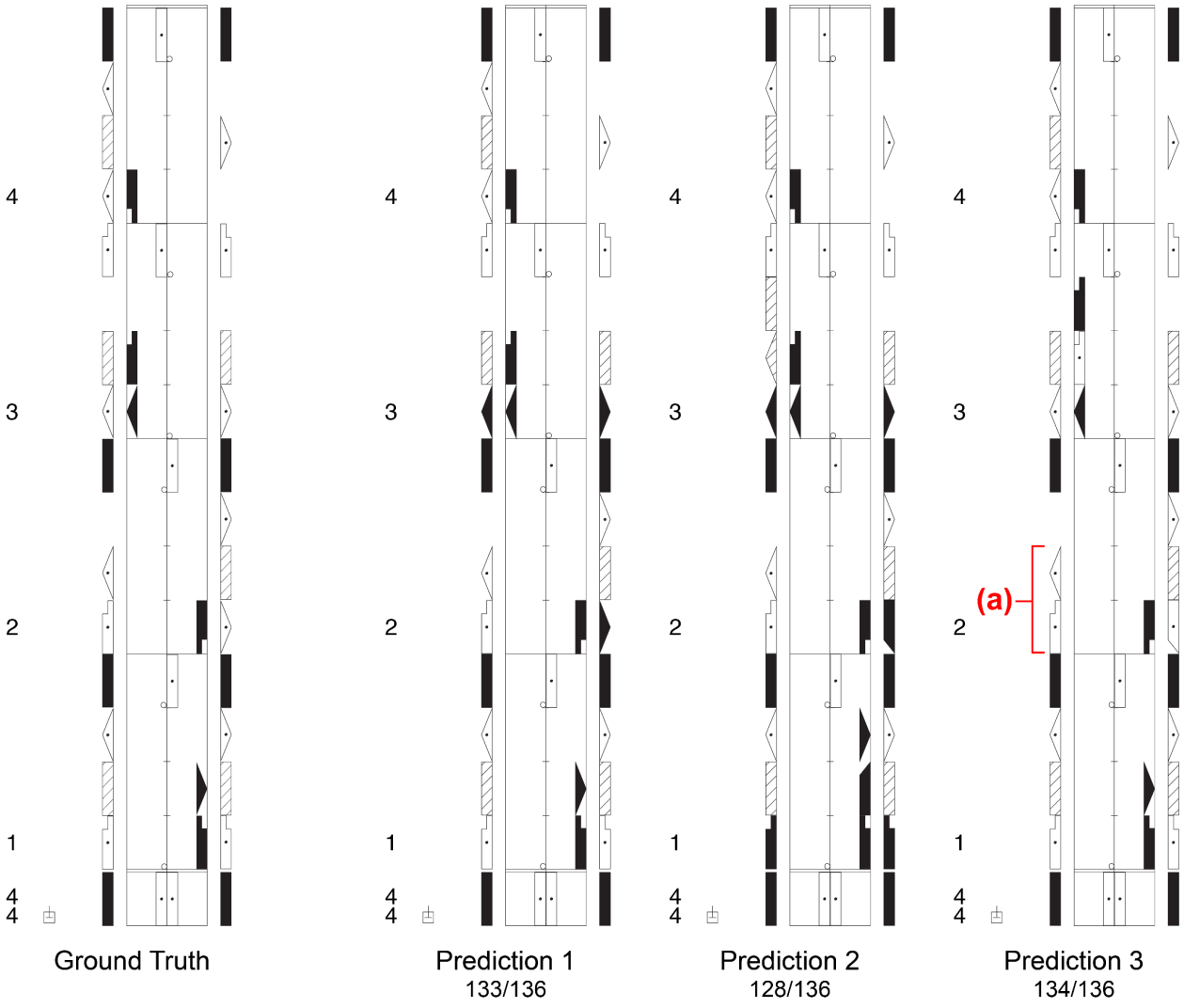
Note. Imprecision in right leg and arm gestures cause incorrect symbols in beats five and six.

These images are mirrored (i.e., my right arm is located at the right side of the image).

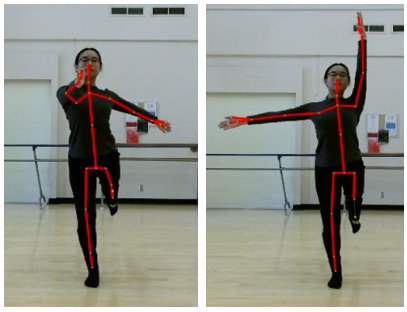
Combination 8: Arms and Legs 2

This combination features the exact choreography of Combination 7 but with the legs at the low level. I wanted to test if the difficulty of maintaining the positions of the legs at the middle level affected my performance of the combination.

Overall Accuracy: 96.8%



(a)



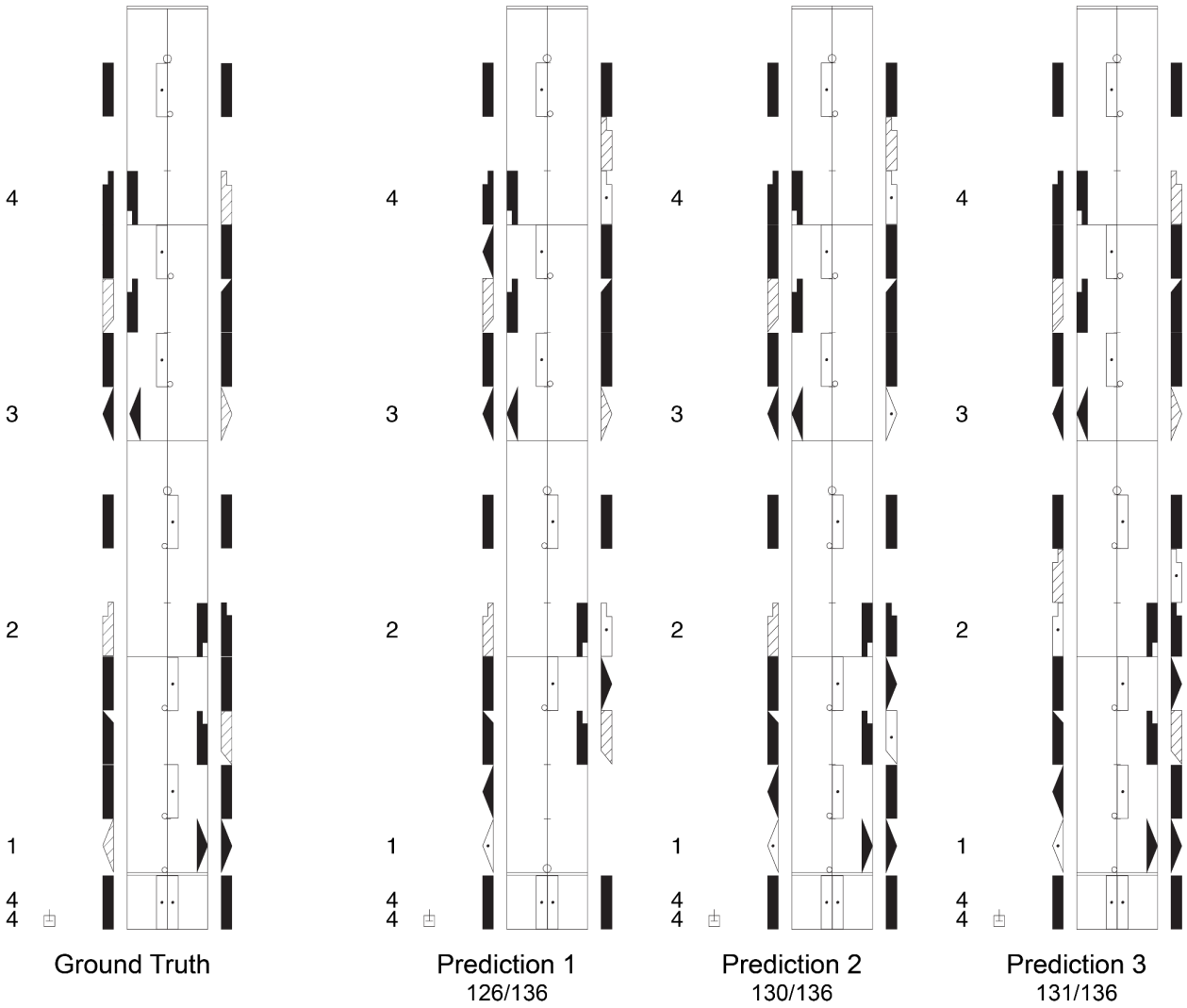
Note. Compared to the previous combination, both arm and leg gestures of beats five and six are executed and predicted more accurately, likely due to the easier nature of the choreography.

These images are mirrored (i.e., my right arm is located at the right side of the image).

Combination 9: Arms and Legs 3

This combination revisits a similar concept of the *dégagé*, with movements that move in and out of place.

Overall Accuracy: 94.9%



Discussion

In the following discussion of the output, I will refer to the combinations as C proceeded by its number (C1, C2, etc.).

Temporal Relationship of Movement

Dance and movement are inherently intertwined with time and tempo. In dance, the individual positions of the body create a complete work only when viewed sequentially in time. Thus, the temporal relationship is significant when interpreting movement. One drawback of my program is that it does not take into account the continuous nature of the movement; it sees each beat or frame as a discrete, independent snapshot of time. As a result, the accuracy of the Labanotation is heavily reliant on the dancer's performance of the combination, and it is highly prone to human error. For example, take the results of C1 and C3, which both assessed arm gestures. The one error in C1 is most likely due to imprecise execution of the movement; my right arm was late to return to place low. In C3, errors occur due to the arm not completely finishing its trajectory of motion by the end of the beat, causing the motion to appear across two beats of the score instead of one beat.

Another recurring instance of this flaw is demonstrated by C4. In the leg gestures, the program often interpreted the side direction as forward diagonal and the backward direction as backward diagonal. An explanation for this is because most people, even trained dancers, do not have perfect 180° turnout, making it difficult to hold the leg directly to the side, and other anatomical limitations such as back flexibility make it difficult to hold the leg directly to the back. This resulted in a position slightly off from the side and backward directions, and the computer classified it as a diagonal direction. A human Labanotator would be able to infer the

complete trajectory of leg movement from front to side to back; however, the computer Labanotator produces the symbol for a beat from only one frame captured by the Kinect, making it incapable of interpolating the full pathway of movement. This is sharply contrasted by the results of C5 which had 100% accuracy. The success of this combination is most likely due to the directness of the motion of a *dégagé.* The legs alternated between place middle in the support and a gesture in a specific direction, leaving no room for ambiguity between movements unlike the circular motions of C4.

While I was recording C7, I noticed a lot of imprecision in the computer predictions of the leg gestures. In this combination, I performed all leg gestures at the middle level, meaning I had to lift and hold my leg at the height of my hip. This is challenging, especially in terms of balance and moving my leg to an exact position at an exact time. I decided to rerecord the combination with the leg gestures at the low level to alleviate some of the unnecessary difficulty; the results are C8. Compared to C7, the previous version of the combination, the accuracy of C8 is improved, suggesting that the difficulty of the choreography–middle versus low level of the leg gesture–did in fact affect the accuracy of the Labanotation because I was able to more precisely execute the movements. Furthermore, the arm gesture accuracy also increased.

In an ideal situation, none of these human factors would affect the results of the Labanotation. When writing Labanotation, humans can make temporal and spatial inferences about the pathway of movement even if the performer is not 100% on time, such as the case of C1 or C3, or if they have imprecise gestures, such as the case of C4 and C7. Teaching a computer to recognize these subtleties would require an even more complex level of analysis across multiple frames of movement and perhaps would benefit from some learning techniques in computer vision and machine learning similar to that of the 3D pose estimation.

Occlusion and Depth Sensing

Occlusion refers to the obscuring of an object behind something else so that the viewer's line of sight to that object is blocked. In the case of 3D pose estimation, self-occlusion occurs when one body part is hidden from the camera behind another body part, such as placing one's arm behind one's own back. For all 3D pose estimation models including the Kinect, occlusion and self-occlusion present a massive challenge. Obdržálek et al. (2012) assessed the robustness of the Kinect in a study examining its use in tracking exercises for elderly people. They conclude about the pitfalls in the Kinect's implementation with respects to occlusions:

> The markerless skeleton tracking of Kinect depends solely on dense depth information and thus frequently fails due to occlusions (e.g. self-occlusion by other body parts, especially if only a single Kinect device is used), non-distinguishing depths (limbs close to the body) or clutter (other objects in the scene, e.g. a chair).

They reference another significant challenge 3D pose estimation models face: depth sensing, or the prediction of joints' z-positions. I will discuss this issue and how the Kinect addresses it in more detail in the next section of this paper.

In C2, the Kinect had difficulty identifying backward low and backward high. Although not as prevalent of an issue, it had difficulty with backward low in C6 as well. This could be due to the fact that it is anatomically difficult to move the arm to a proper, 45° backward low and backward high. However, it may also be due to the fact that the backward direction relies on an accurate estimation of the z-position by the Kinect. Furthermore, in the backward direction, the occlusion of the gesturing limb is much more likely than in any other direction. The inaccuracy most likely stems from a combination of all of these issues. Unfortunately, this is a shortcoming

of 3D pose estimation technology in its current state, but as it is a recurring area of interest, improvements are made as technology in this field progresses.

Locomotion, Place Low Leg Gesture, and Place Middle Support

Another important limitation of this study was that I was unable to delve into the analysis of locomotion. Locomotion is the movement in space from one location to another, which is indicated in Labanotation by symbols in the support column. In order to tackle the analysis of locomotion, a few changes to the characteristics of the program as it is currently would be critical. First, as discussed earlier, an algorithm that considers the temporal relationship of movement is essential to deducing whether a person has traveled in space from one beat to the next.

Second, the program must be able to differentiate between leg gestures and supports. A leg is a support when it carries some weight of the body (i.e., standing on the leg) whereas a leg gesture occurs when there is no weight on that leg (i.e., lifting it off of the floor). As is, the program can only detect place middle for the legs in the support column, which overrides its ability to recognize place low in the gesture column. The difference between place middle support and place low gesture is subtle; in both positions, the leg is directly underneath the corresponding hip. In place middle support, the person is standing on that leg. With a place low gesture, the leg is hovering just slightly off of the ground so that there is no weight on it. This dilemma raises the question of how a computer would be able to recognize whether the leg is supporting or gesturing at all in any direction. My present algorithm of calculating the angle of the limb would fail this task and necessitates a more robust system that can either discern where the ground is located from the frame or is calibrated before to know where the ground is before

even beginning the pose estimation. Further details such as the level of the support demand a degree of detail and precision that simple pose estimators like the Kinect cannot achieve.

Other Observations about 3D Pose Estimation Models

Prior to using the Microsoft Kinect, I reviewed and tested various 3D pose estimation models that only required RGB images as input.[8] For the ones I was able to run, the models proved to be too inaccurate to generate Labanotation. One explanation for the overall inaccuracy of the models is that the models were trained on datasets of pedestrian body positions, not positions more commonly performed by dancers. In machine learning, a model can only identify data similar to what it has learned through training on datasets. Like a newborn that has not yet experienced much of the world, it cannot recognize sights it has not seen before. However, unlike the human brain, pose estimation models are not advanced enough to accurately extrapolate information from images they have already seen, at least not in the same way humans perceive patterns. After training on images of people performing everyday tasks, it has difficulty detecting people performing more specialized tasks. A human would be able to identify another human regardless of the activity they are doing with much more ease.

To illustrate, take this example of non-pedestrian movement. When recording testing videos, I only performed movements with straight arms and legs. While this condition was set in order to simplify the Labanotation, it may have caused difficulty with the pose estimation. Rarely do people stand with completely extended arms and legs, and even more rarely do we move our arms and legs without bending them. A pose estimation model, for instance, may have learned to identify an arm by detecting its shoulder and elbow joints. Without bending the arms, it may not properly locate the elbow, thus causing errors in the detection of the arm.

[8] A more in-depth, technical discussion of these models and methods of 3D pose estimation are in the Appendix.

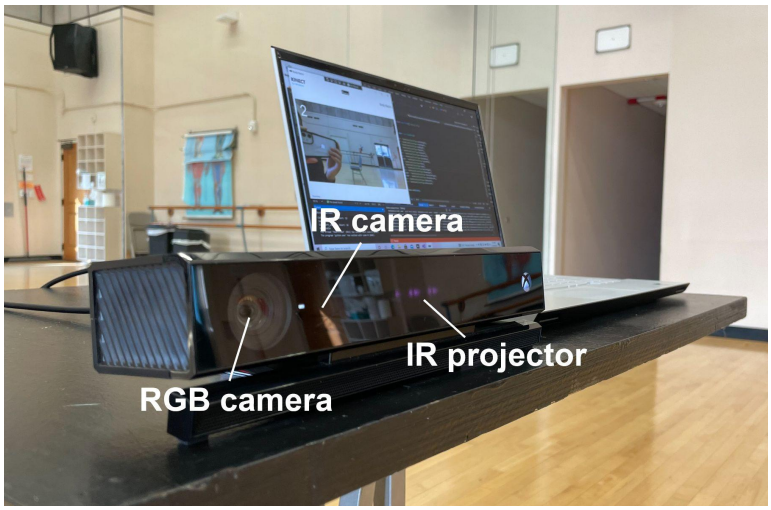
A solution for this disparity between natural pedestrian movement and less conventional dancer movement could be transfer learning. In transfer learning, a pre-trained model is specialized for a more specific instance of or a similar task to what was originally intended by training it further on additional data. In my case, I could generate a dataset of images of people performing movements more commonly found in dance and manually label each image for the correct joints. However, generating and labeling a large enough dataset of thousands of images and training my own model is out of the scope of this thesis.

V. The Microsoft Kinect

3D pose estimation is a significantly more difficult task than 2D pose estimation because of the additional measurement of depth. In a 2D RBG image, the width (x-axis) and height (y-axis) are intuitively set up by the 2D nature of the input image. However, 3D pose estimation models must accurately achieve the tall order of inferring the z-axis that does not exist in a 2D image. The Microsoft Kinect sensor provided a relatively easy, cost-effective solution to this challenge. First introduced in 2010 as an accessory for the Xbox 360, Microsoft released the Kinect 360 with the goal of replacing hand-held controllers, like that of the Wii, in the task of detecting the player's movements. In addition to a typical RGB camera and a four-microphone array, the Kinect sensor contains a depth sensor that consists of an infrared (IR) projector and IR camera (Figure 10). The IR projector emits a speckle dot pattern of light that is only visible to the IR camera. Using a technique called structured light, the Kinect 360 is able to geometrically reconstruct the depth at the location of each IR dot (Zhang, 2012).

With the calculated depth information, the Kinect then can perform 3D pose estimation. Starting with 2D classification as an intermediate step, it categorizes each pixel of the frame into

Figure 10



Note. The components of the Kinect v2.

a body part. With these clusters of pixels, it finds a "global centroid," or an average point in the middle of each body part to estimate the location of each joint. Lastly, the joints of the 3D pose estimation are finalized using additional information to ensure the body position is realistic (Zhang, 2012).

Following its release, the Kinect 360 garnered substantial attention amongst developers who were interested in its potential applications in computer vision. As opposed to using extremely expensive, cumbersome motion capture technology that requires a particular indoor setup, the Kinect is portable and costs only a few hundred dollars for comparable pose estimation capabilities. Various companies created unofficial software development kits (SDKs) to assist developers with the incorporation of the Kinect into their projects. In 2011, Microsoft released an official Kinect SDK, as well as the OpenKinect and OpenNI libraries (Cai et al., 2016).

In 2014, Microsoft introduced the Kinect for Xbox One, commonly referred to as the Kinect v2. This second-generation Kinect proved to be more accurate than the Kinect 360, due to

its use of time of flight technology instead of structured light to perform depth estimation. In time of flight, the Kinect v2 uses the same infrared speckle dot pattern but instead measures the time it takes for a dot to be projected from the IR projector onto the scene and reflected back into the IR camera (DiFilippo and Jouaneh, 2015). Other updates from the Kinect 360 to the Kinect v2 include a 1920x1080 resolution RGB camera and 512x424 depth camera, the use of 25 joints (as opposed to 20) for pose estimation, and the ability to detect up to 6 people per frame (Cai et al., 2016). For this thesis, I used the Kinect v2 with the adapter for Windows PC and the Kinect SDK 2.0. The Kinect SDK 2.0 provides a variety of sample code for Kinect features including 3D pose estimation, hand and facial pose estimation, speech recognition, and gesture detection. The 3D pose estimation code operated as the starting point for the code in this thesis concerning the Kinect.

By 2017, Microsoft completely phased out the production of the Kinect; now, it can only be purchased by resale. Despite the enthusiastic reception within the first couple years of its release, the Kinect experienced a steep downfall in popularity likely due to the lack of applications developed around it (Thier, 2018). Without applications, Microsoft could not prove the practicality of the Kinect in the long term, and the novelty of a fancy, depth-detecting gadget soon wore off. Furthermore, developers found that the depth sensor was not as accurate as it was initially thought to be, thus taking another toll on its usability in practical applications (Cai et al., 2016). A reoccurring opinion is that the Kinect was simply too advanced for its time and could not live up to the expectations that Microsoft created for it. The Microsoft Kinect has since been replaced by the Azure Kinect, a more timely, and accurate, iteration of a Microsoft depth sensor.

CONCLUSION

3D human pose estimation is still a fledgling area of study. Additional areas of development that I did not elaborate on earlier includes multi-person pose estimation, one step further from single-person pose estimation. In fact, the Kinect v2 is capable of predicting the positions of up to six different people within the same frame. With more people, however, there is drastically increased variability, such as from occlusions that result from having a more crowded frame, and the computation power necessary to perform this task grows with the number of people in the image (Wang et al., 2021). This is just one example of the many ways researchers continue to push the boundaries on the current state of pose estimation.

When paired with dance, the opportunities are unimaginable. A recent interview published by the Dance Notation Bureau highlights the work of Katsu Ikeuchi and Machiko Sato in teaching a robot how to dance through interpreting Labanotation (Russ, 2021). According to Ikeuchi, the primary motivation in this project is using "robotics theory to gain a deeper understanding of [Japanese cultural] dances in addition to preserving them." Through the courses I took with the Dance Notation Bureau, I discovered that Labanotation concisely captures the intricacies of dance that video fails to retain. In cultural preservation projects like that of Ikeuchi and Sato and the researchers from Beijing Jiaotong University, Labanotation provides that utmost level of complexity that is crucial to preserving the integrity of the movement. Utilizing technology alongside it allows the unique meticulousness of Labanotation to be accessible to the larger population of dancers, or even non-dancers in general, who have not received the detailed training necessary to read Labanotation.

In an age where technology like virtual and augmented reality make the holograms of sci-fi movies reality, dance, just like any other field, has the chance to evolve and progress.

Although nothing surpasses live, in-person dance, I can see a future where technology aids in live performances to create novel experiences for audiences. Furthermore, motion capture or pose estimation technology could play a significant role in injury prevention or physical therapy. Neither dance nor technology should exist in a vacuum, and their greatest potential is reached when innovative minds collaborate to create these types of interdisciplinary advancements.

BIBLIOGRAPHY

Cai, Z., Han, J., Liu, L., & Shao, L. (2016). RGB-D datasets using Microsoft Kinect or similar sensors: A survey. *Multimedia Tools and Applications*, *76*(3), 4313–4355. https://doi.org/10.1007/s11042-016-3374-6

Chang, J. Y., Moon, G., & Lee, K. M. (2018). V2V-posenet: Voxel-to-voxel prediction network for accurate 3D hand and human pose estimation from a single depth map. *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 5079–5088. https://doi.org/10.1109/cvpr.2018.00533

Choensawat, W., Nakamura, M., & Hachimura, K. (2014). Genlaban: A tool for generating labanotation from Motion Capture Data. *Multimedia Tools and Applications*, *74*(23), 10823–10846. https://doi.org/10.1007/s11042-014-2209-6

Choensawat, W., Nakamura, M., & Hachimura, K. (2015). Applications for recording and generating human body motion with labanotation. *Dance Notations and Robot Motion*, 391–416. https://doi.org/10.1007/978-3-319-25739-6_19

DiFilippo, N. M., & Jouaneh, M. K. (2015). Characterization of different Microsoft Kinect sensor models. *IEEE Sensors Journal*, *15*(8), 4554–4564. https://doi.org/10.1109/jsen.2015.2422611

Gong, W., Zhang, X., Gonzàlez, J., Sobral, A., Bouwmans, T., Tu, C., & Zahzah, E.-hadi. (2016). Human pose estimation from monocular images: A comprehensive survey. *Sensors*, *16*(12), 1966. https://doi.org/10.3390/s16121966

Guest, A. H. (2005). *Labanotation: The System of Analyzing and Recording Movement* (4th ed.). Routledge.

Guo, H., Miao, Z., Zhu, F., Zhang, G., & Li, S. (2014). Automatic labanotation generation based on Human Motion Capture Data. *Communications in Computer and Information Science*, 426–435. https://doi.org/10.1007/978-3-662-45646-0_44

Krenker, A., Bester, J., & Kos, A. (2011). Introduction to the artificial neural networks. *Artificial Neural Networks - Methodological Advances and Biomedical Applications*, 1–18. https://doi.org/10.5772/15751

Liu, Z., Zhu, J., Bu, J., & Chen, C. (2015). A survey of human pose estimation: The body parts parsing based methods. *Journal of Visual Communication and Image Representation*, *32*, 10–19. https://doi.org/10.1016/j.jvcir.2015.06.013

Mehta, D., Sridhar, S., Sotnychenko, O., Rhodin, H., Shafiei, M., Seidel, H.-P., Xu, W., Casas, D., & Theobalt, C. (2017). VNect: Real-time 3d human pose estimation with a single rgb camera. *ACM Transactions on Graphics*, *36*(4), 1–14. https://doi.org/10.1145/3072959.3073596

Munea, T. L., Jembre, Y. Z., Weldegebriel, H. T., Chen, L., Huang, C., & Yang, C. (2020). The progress of human pose estimation: A survey and taxonomy of models applied in 2D human pose estimation. *IEEE Access*, *8*, 133330–133348. https://doi.org/10.1109/access.2020.3010248

Obdržálek, Š., Kurillo, G., Ofli, F., Bajcsy, R., Seto, E., Jimison, H., & Pavel, M. (2012). Accuracy and robustness of Kinect pose estimation in the context of coaching of elderly population. *2012 Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, 1188–1193. https://doi.org/10.1109/embc.2012.6346149

Ratner, A., Varma, P., Hancock, B., & Ré, C. (2019, March 10). *Weak supervision: A new programming paradigm for Machine Learning*. The Stanford AI Lab Blog. http://ai.stanford.edu/blog/weak-supervision/

Russ, H. (2021, Fall). Dancing robots: An interview with Katsu Ikeuchi and Machiko Sato. *Library News From the Dance Notation Bureau.*

Thier, D. (2018, January 4). *Microsoft's Xbox Kinect is now really, truly dead*. Forbes. https://www.forbes.com/sites/davidthier/2018/01/04/microsofts-xbox-kinect-is-now-really-truly-dead/?sh=6d334fcd1195

Tome, D., Russell, C., & Agapito, L. (2017). Lifting from the deep: Convolutional 3D pose estimation from a single image. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2500–2509. https://doi.org/10.1109/cvpr.2017.603

Topaz, M. (1996). *Elementary Labanotation: A Study Guide*. Dance Notation Bureau Press.

Wang, J., & Miao, Z. (2018). A method of automatically generating labanotation from Human Motion Capture Data. *2018 24th International Conference on Pattern Recognition (ICPR)*, 854–859. https://doi.org/10.1109/icpr.2018.8545306

Wang, J., Tan, S., Zhen, X., Xu, S., Zheng, F., He, Z., & Shao, L. (2021). Deep 3D human pose estimation: A review. *Computer Vision and Image Understanding*, *210*, 103225. https://doi.org/10.1016/j.cviu.2021.103225

Wilke, L., Calvert, T., Ryman, R., & Fox, I. (2003). Animating the Dance Archives. *VAST*, 93–100. https://doi.org/10.2312/VAST/VAST03/093-100

Wilke, L., Calvert, T., Ryman, R., & Fox, I. (2005). From dance notation to human animation: The Labandancer Project. *Computer Animation and Virtual Worlds*, *16*(3-4), 201–211. https://doi.org/10.1002/cav.90

Xie, N., Miao, Z., Wang, J., & Zhang, Q. (2020). End-to-end method for labanotation generation from continuous motion capture data. *2020 IEEE International Conference on Multimedia and Expo (ICME)*, 1–6. https://doi.org/10.1109/icme46284.2020.9102860

Zhang, H.-B., Lei, Q., Zhong, B.-N., Du, J.-X., & Peng, J. L. (2015). A survey on human pose estimation. *Intelligent Automation & Soft Computing*, *22*(3), 483–489. https://doi.org/10.1080/10798587.2015.1095419

Zhang, Z. (2012). Microsoft Kinect sensor and its effect. *IEEE Multimedia*, *19*(2), 4–10. https://doi.org/10.1109/mmul.2012.24

Zhou, X., Huang, Q., Sun, X., Xue, X., & Wei, Y. (2017). Towards 3D human pose estimation in the wild: A weakly-supervised approach. *2017 IEEE International Conference on Computer Vision (ICCV)*, 398–407. https://doi.org/10.1109/iccv.2017.51

Zhou, Z., Miao, Z., & Wang, J. (2016). A system for automatic generation of labanotation from Motion Capture Data. *2016 IEEE 13th International Conference on Signal Processing (ICSP)*, 1031–1034. https://doi.org/10.1109/icsp.2016.7877986

APPENDIX

As part of this thesis, I explored methods of 3D pose estimation that do not rely on specific hardware such as the Microsoft Kinect. Predicting 3D positions or depth information from only a 2D input (e.g., an image or a video) is a well-researched area of study with many diverse ways of achieving this. In a recent paper from 2021, Wang et al. outline the current popular approaches to solving this problem of 3D pose estimation. It is important to acknowledge that there are different ways to model the body in human pose estimation; in this appendix, I will focus on the skeletal model of connecting joints to represent the human body, which I have been using for the entirety of this thesis. Wang et al. (2021) split their discussion into 3D pose estimation using a 2D *frame* and 3D pose estimation using a 2D *sequence of frames*. In other words, they examine 3D pose estimation from a photo versus from a video. Another distinction the authors make is the option of using a *monocular* image versus *multi-view* images. This has to do with the number of cameras in use; monocular means there is only one camera and thus one stream of input frames, whereas multi-view means there are multiple cameras and thus multiple views to base the pose estimation on. Within this summary of current methods, I also reference various 3D pose estimation models I investigated throughout the thesis as examples of these methods.

One frame, one camera. In monocular pose estimation from a single frame, two main strategies are the most common. One strategy is to directly estimate the 3D joints from the 2D image. Directly estimating 3D joints can be performed by detection-based methods, where the location of each joint is predicted independently, or by regression methods, where the location of each joint is predicted in relation to a "root" joint. As an example, Chang et al. (2018) utilize what they call "voxel-to-voxel prediction" by converting the 2D image to voxels, a way to

represent 3D points, and executing the pose estimation completely in 3D. The problem with these methods is that they are highly costly in terms of computation space and time; they require a very powerful computer and a lot of time. Another approach that saves some of this computation cost is called “lifting” from 2D to 3D. The model first executes a less costly 2D estimation of the joints as an intermediate step and projects the 2D to 3D during the last step of a convoluted neural network (CNN), for example. Tome et al. (2017) use lifting from 2D to 3D as a recurring step in their CNN to increasingly raise the accuracy of the pose estimation.

One frame, multiple cameras. An alternative method to monocular pose estimation is using multi-view images, which provides additional information to base the pose estimation on. One method is combining multiple 2D joint heatmaps using constrained optimization. First, the model performs 2D pose estimation on each of the camera views; then, it merges all of the 2D predictions into a single, 3D pose estimation. Additionally, a model can be trained with weak supervision, utilizing multiple camera views during training. Weak supervision is a technique in machine learning where “noisier” datasets are used in place of large sets of precise, hand-labeled data to train models; these labeled dataset are costly and time-consuming to create and often implausible with the given resources (Ratner et al., 2019). An example of this is the model by Zhou et al. (2017), which I used in the initial stages of this project. This model uses weak supervision to train with sequences of images with both 2D and 3D labels. Another use of multi-view images is triangulation, a traditional method of 3D reconstruction, but it requires tremendous amounts of geometrical computation.

Multiple frames, one camera. When the input is multiple, consecutive frames of movement (i.e., a video), the pose estimation model can utilize information about time and continuity of motion. The format of this input can be a prerecorded video or real-time input,

where the pose estimation model processes the frames as they are recorded. For sequences of monocular images, the same methods used for single frames of monocular images apply (see above); however, there are additional calculations that take into consideration temporal aspects. One technique called Long Short-Term Memory (LSTM) trains a model to learn about the "temporal relationship between poses" (Wang et al., 2021). In other words, the model looks at a large amount of data to create conclusions about human motion. Is it realistic for a human to move their arm from point A to point B in X amount of time? What are common patterns of movement? With this type of knowledge, the model can make more accurate predictions by penalizing or avoiding unrealistic sequences of pose estimations. One instance of real-time, single-view 3D pose estimation is VNect, created by Mehta et al., where they fit joint predictions to a "kinematic skeleton," prioritizing the temporal consistency of poses.

Multiple frames, multiple cameras. Obtaining sequences of frames from multiple camera views is a more recent expansion in pose estimation, and the idea is to integrate the advantages of all of the previously mentioned methods. However, due to the massive amount of data collected from this type of setup, it requires a very precise, controlled environment for reasonable results.