David Johnson, Daniela Damian, and George Tzanetakis
Department of Computer Science
University of Victoria
3800 Finnerty Road
Engineering and Computer Science
Building, Room 504
Victoria, BC V8P 5C2 Canada
davidjo@uvic.ca, danielad@uvic.ca,
gtzan@ieee.org

Detecting Hand Posture in Piano Playing Using Depth Data

Abstract: We present research for automatic assessment of pianist hand posture that is intended to help beginning piano students improve their piano-playing technique during practice sessions. To automatically assess a student's hand posture, we propose a system that is able to recognize three categories of postures from a single depth map containing a pianist's hands during performance. This is achieved through a computer vision pipeline that uses machine learning on the depth maps for both hand segmentation and detection of hand posture. First, we segment the left and right hands from the scene captured in the depth map using per-pixel classification. To train the hand-segmentation models, we experiment with two feature descriptors, depth image features and depth context features, that describe the context of individual pixels' neighborhoods. After the hands have been segmented from the depth map, a posture-detection model classifies each hand as one of three possible posture categories: correct posture, low wrists, or flat hands. Two methods are tested for extracting descriptors from the segmented hands, histograms of oriented gradients and histograms of normal vectors. To account for variation in hand size and practice space, detection models are individually built for each student using support vector machines with the extracted descriptors. We validate this approach using a data set that was collected by recording four beginning piano students while performing standard practice exercises. The results presented in this article show the effectiveness of this approach, with depth context features and histograms of normal vectors performing the best.

Assessment of Piano Technique

Learning to play a musical instrument is a challenging task that requires years of disciplined practice to master. Typically, aspiring musicians rely on weekly lessons with a professional teacher to supervise and provide feedback on their learning progress. To improve their playing abilities, students must augment weekly lessons with daily practice in which they are expected to gradually be able to self-analyze their performance. Students must then wait for their next lesson to receive expert feedback on their practice and technique. Some teaching methods, such as the Suzuki method, expect involvement from student's parents to actively supervise and provide feedback during the daily practice of their child. This is often challenging or infeasible for busy parents, and such an approach cannot be easily transposed for adult students. The ubiquity of computers as well as the emergence of virtual reality and advances in sensor technology provide new research opportunities to create innovative tools to assist both students and teachers with the process of learning a musical instrument.

Research in computer-assisted music-instrument tutoring (CAMIT) systems attempts to enhance music pedagogy by providing the tools necessary to automatically assess student performance and provide personalized feedback (Percival, Wang, and Tzanetakis 2007). Many CAMIT systems rely on sound analysis and are said to be "listening" to students' performances during practice. Thus, the feedback students receive accounts for the musical quality of their performance, omitting evaluation and feedback of their physical playing technique. CAMIT researchers have also recognized the importance of assessing physical technique to improve musical performance. Projects such as i-Maestro (Ng, Nesi, and Marta 2008) and Technology Enhanced Learning of Music Instruments (http://telmi.upf.edu) have implemented methods for the automatic assessment of technique in stringed instrument practice. The major contribution of our work is a system for the automatic assessment of piano playing technique by

Computer Music Journal, 43:1, pp. 59–78, Spring 2019
doi:10.1162/COMJ_a_00500

"watching" a student's hands, rather than relying on acoustic analysis, to provide feedback on their hand posture. The system is intended to be used to generate feedback for a tutoring interface that would augment weekly piano lessons. Such an interface would enhance the piano-learning process by providing students with immediate feedback on their performance during practice. Furthermore, automated feedback would allow teachers to track students' progress during practice sessions in which the teacher is not present.

This work builds on our previous research on hand-posture detection, in which we implemented a prototype detection system, trained and tested with two experienced adult piano players (Johnson et al. 2016). Because the participants were experienced and generally played with correct posture, the data used had to be artificially created by asking the participants to play a set of exercises with both correct and deliberately incorrect postures. Although this initial work showed the potential for hand-posture detection, it was not tried out with piano players for whom the different hand postures occurred naturally. The work presented in this article extends the research in two ways: (1) by analyzing empirical data from a real-world data set captured by video-recording real piano students and (2) provides a revised, more robust hand-segmentation method and a new posture-detection training scheme that we developed to analyze the smaller hands of the piano students in the new data set. Results of experiments on the new approaches show that they are more robust and work with piano students of varying ages.

Pianist Hand Posture

Body and hand posture are fundamental to proper technique in piano playing. Riley, Coons, and Marcarian (2005) discuss the importance of performance feedback in the acquisition of musical skill, especially in the case of repetitive practice in which consistent bad technique may lead, in extreme cases, to injuries. In their work, multimodal feedback of pianist technique was generated through analysis of MIDI data, video recordings, and surface electromyography. The latter was added to augment video analysis after the authors found that, even for experienced pianists, reviewing videos frame by frame did not help identify problems. Augmenting the system with surface electromyography improved the results, but the authors noted that analysis required time and patience from both the student and the instructor.

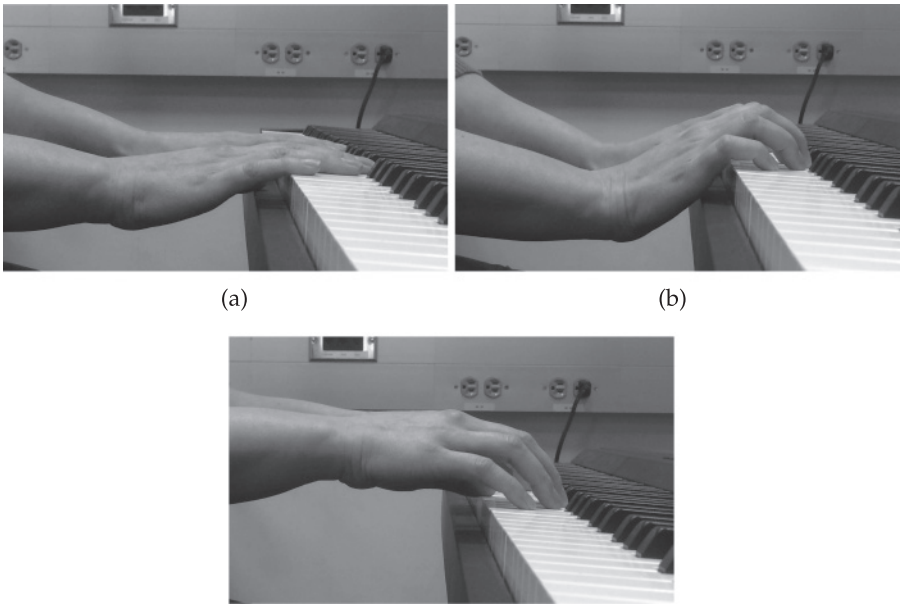
In contrast, our system is intended to generate data for immediate performance feedback without complex analysis.

For correct hand posture, the hand should be arched and the fingers curled as illustrated in Figure 1c. Working with a piano teacher, we identified two common posture mistakes observed in students: playing with flat hands (Figure 1a), and playing with low wrists (Figure 1b). Because most of a student's practice time occurs between lessons, bad habits can quickly become chronic. Providing students with a tool that can identify and help correct these mistakes during daily practice would reduce the probability that they become ingrained in the student's technique.

Related Work

To improve musical abilities, students must learn to self-evaluate their performance to identify and correct errors during practice sessions. With teacher-based training, students rely on their teachers to point out errors and provide them with instruction to address the errors. Students, however, have limited practice time with teachers and may have difficulty remembering everything learned during a teacher-led session. To improve the learning process, CAMIT systems attempt to help students evaluate their performance using computational techniques for automatic assessment when a teacher is not present. In general, errors can be categorized into musical mistakes, such as missed notes or poor sound quality, and technique mistakes, such as poor posture. There has been research into CAMIT systems for both categories.

Figure 1. The three common hand postures of beginning piano students that are detected with the presented system: flat hands (a) and low hands (b) are common postures mistakes made by students, as opposed to the hand in the ideal posture for pianists (c).



(a) (b)

(c)

Musical Evaluation

Evaluation of the musical component of a performance is usually done by listening to the performance to identify musical errors e.g., incorrect notes or poor sound quality). To assess musical performance, CAMIT systems often use audio signal processing (ASP).

One of the first CAMIT research projects to use ASP was the Piano Tutor project (Dannenberg et al. 1993), an intelligent multimedia system to teach beginners to play the piano. The Piano Tutor was a complete tutorial system intended to supplement traditional music pedagogy by a professional teacher. Using ASP, the Piano Tutor implemented score following to assess how a student was performing by listening to the student's performance and comparing it with a score (Dannenberg et al. 1990).

In the project IMUTUS (Interactive Music Tuition System, cf. Raptis et al. 2005) a music tutoring system was developed for teaching the recorder to beginning students. Like the Piano Tutor, IMUTUS listened to student performances using ASP for audio recognition to assess the musical output. Audio recognition was paired with score matching to detect errors in the performance. By listening to a performance, IMUTUS was able to detect melodic, timing, and articulation errors (Schoonderwaldt, Askenfelt, and Hansen 2005).

The Interactive Digital Violin Tutor (iDVT, cf. Lu et al. 2008) was a system for violin tutoring that transcribed a student's performance through onset detection and pitch estimation. To improve the quality of onset detection, ASP was fused with video data. A student could then compare the transcribed performance to a reference score.

Research performed with the TELMI project also used ASP to analyze violin performance but, rather than focus on pitch and onset errors, the authors used audio data to assess tone quality using machine learning (Giraldo et al. 2019). The system implements methods to build user-defined tone quality models to overcome the subjectivity in timbre perception that makes generalization a challenge. Audio signal processing plays an important role in the automatic assessment of musical performances, but it can only assess the musical quality of the performance. Other methods are needed to assess performer technique, such as pianist hand posture.

Evaluation of Technique

Automatic assessment of technique requires methods for capturing body position and movements during practice. Researchers in CAMIT have used optical systems, such as motion-capture systems and camera technologies, to capture the needed performance data.

For piano pedagogy, Mora and colleagues (2006) used a motion-capture system to track the movements and body posture of a pianist. The system used eight infrared cameras and an average of 79 positional markers to record positional data to construct a 3-D skeleton model that could be overlaid on a video recording of the practice session.

The i-Maestro project (Ng et al. 2007) used a motion-capture system to capture and analyze performance on stringed instruments for the three-dimensional (3-D) augmented mirror application. Twelve infrared cameras and markers attached to the performer, the bow, and the instrument were used in the augmented mirror to capture performer and instrument positional data. The data were used to provide assessment and feedback on the performer's bowing technique and posture.

Motion-capture systems, however, are complicated and expensive, limiting their use outside of laboratory settings. Thus, more accessible methods, such as computer vision or signal processing with low-cost sensors, are needed to capture motion for technique assessment.

Dalmazzo and Ramirez (2019) used the Myo armband, which tracks muscle movement in the forearm using electromyography, for the classification of violin bowing gestures. The Myo data were combined with audio data for real-time gesture recognition using a hierarchical hidden Markov model.

Salgian and Vickerman (2016) proposed a computer vision–based CAMIT system for conducting students that used the Microsoft Kinect to track students' physical performance. Using Kinect data, the system was able to detect common conducting errors, calculate tempo, and perform articulation recognition.

These works show that assessment of playing technique is an important component of music pedagogy and can be integrated in CAMIT systems using technologies such as motion capture, computer vision, and ASP. Capturing pianists' hands for assessment of piano technique using these technologies presents its own challenges.

Pianist Hand Tracking

Identifying and tracking pianists' hands for performance analysis has been explored in previous research. Tits and coworkers (2015) used a marker-based motion-capture system to analyze pianists' hands and finger gestures to determine the performer's level of expertise.

Unfortunately, marker-based approaches are generally intrusive and not readily available outside the laboratory environment. As an alternative, markerless approaches for hand tracking use standard RGB cameras or depth maps from depth cameras, such as the Kinect. A *depth map* is an image in which each pixel represents a distance from the depth camera to a point on the depicted object's surface. Hadjakos, Lefebvre-Albaret, and Toulouse (2009) presented three methods for hand assignment using RGB video to detect which hand played a note, and Oka and Hashimoto (2013) used a combination of depth recordings from a Kinect and information from MIDI data to identify a pianist's fingering mistakes. Aristotelis Hadjakos (2012) used a depth camera to capture the motion of key points from a pianist's entire body, such as head, shoulders, wrists, and hands. Finally, Liang et al. (2016) used a depth camera and machine learning to detect finger tapping for playing a virtual piano. These works presented various methods for capturing pianists' hands, but none provided the data needed to analyze hand posture during performance.

There has been some research on systems that capture the precise details of the hand needed for hand-posture analysis. MacRitchie and McPherson (2015) developed a system for automatic fingering detection that fused data from a high-speed camera and touch sensors. A camera placed at an aerial viewpoint tracked painted markers on the pianist's hands to capture the x and y coordinates of each finger. Although the data was only 2-D, the coordinates were used to calculate a curvature index,

Figure 2. The depth camera is positioned with an aerial viewpoint to capture both hands from overhead (a). (Note that in the final system we used an Intel Realsense SR300 in place of the Kinect displayed here.) Overhead RGB view of the camera which is used for data annotation (b). Example of a depth map that is used for model training and detection (c).

(a) (b) (c)

CI. This index was calculated as the ratio between the distance of two points at a given time with distance of the same two points in a reference frame. Although this provides relative information about the curvature of each finger, there is not enough information to fully discriminate between various categories of hand posture. Li et al. (2014) proposed a system for pianist hand-posture analysis that detected key regions of the hand using 3-D data. The authors used computer vision with depth cameras to find regions of the hand, such as the hand center, the middle finger, and the wrist. The key points were used to derive features for analysis: the ratio of hand-center height to hand-arch height, and the horizontal and vertical wrist angles. Using these features, a histogram analysis was performed for assessing the range of hand motion during a specific piano piece. The histograms used for analysis were generated using data from the entirety of a performed piano piece and were not used for real-time classification of posture mistakes.

System Description

To enhance the piano pedagogy process, students need feedback on their physical performance in addition to feedback about their musical performance. As previously discussed, there is little work being done in systems that do more than just listen to students perform. In this article, we present a novel approach for watching students practice to detect hand-posture mistakes using a 3-D camera. The use of a this type of camera allows for a noninvasive setup and easy installation in any practice space.

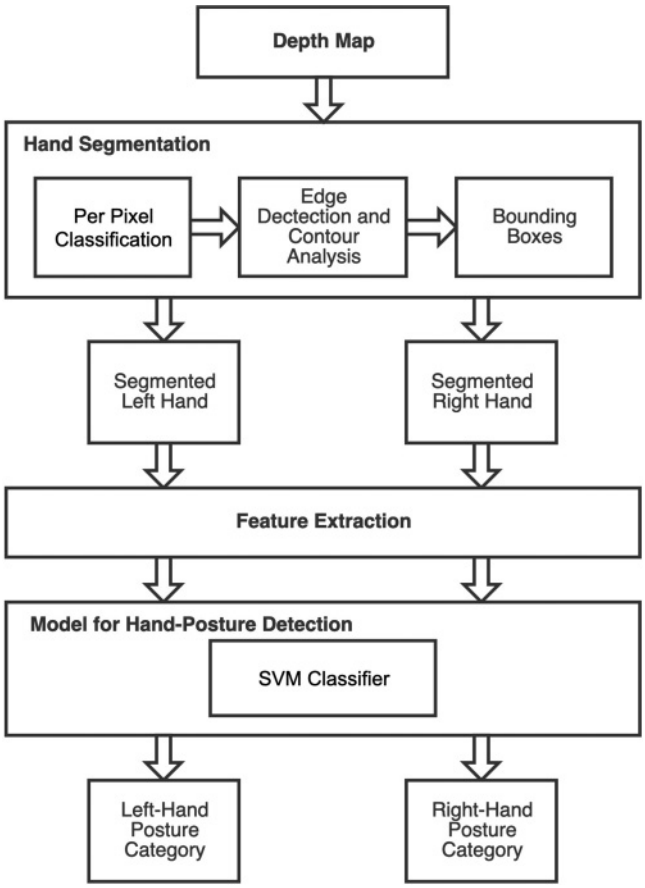
The posture-detection system uses a 3-D structured-light camera, such as a Kinect or Intel Realsense SR300, placed above the piano to capture a piano student's hands from above. Figure 2 shows the camera placement (Figure 2a), the scene captured by the camera (Figure 2b), and an example of a depth map as recorded by an SR300 (Figure 2c). With this configuration, both hands are recorded with a single camera and the depth data are used to capture information about the geometry of the hands, which is used to infer hand posture.

To infer hand posture from a single depth map we propose the image-processing pipeline shown in Figure 3. The first step in the pipeline is the hand segmentation, in which the left and right hands are individually identified in the depth map using per-pixel classification. The results are two masks, one for the right hand and one for the left. Edge detection with contour analysis is performed on the masks to identify the bounding regions of each identified hand. This results in two depth maps containing each hand. Feature descriptors are then extracted from the segmented hands and used for training the hand-posture detection model. Once the model has been trained, the extracted features are used with the model to predict the posture category of the segmented hands.

Data Collection

Data were collected using an Intel Realsense SR300 depth camera. The SR300 uses a short-range structured light system to measure three-dimensional shape at a resolution of 640×480 pixels. It additionally has a 1080i RGB camera. The camera is capable

Figure 3. The posture-detection pipeline used for inferring hand posture from depth maps.



of providing synchronized color, depth, and infrared data at up to 60 frames per second (fps) with depth range of 0.2 to 1.5 m (Carfagni et al. 2017).

Using a data-driven approach for hand segmentation and posture analysis requires a diverse set of data to ensure generalization for our models. As this system is being initially designed for beginners, we recruited piano students between the ages of 9 and 12 years for data collection. Using the SR300, we recorded the students playing a variety of piano exercises. The exercises range from basic scales to technical exercises from the popular piano lesson book series *A Dozen a Day* (Burnam 2005). For each recording we captured the depth data and color data at 30 fps and a resolution of 640 × 480. The algorithms described through this work exploit only the depth information. Color data were used to generate hand masks for training the segmentation model and for human annotation of hand posture used to train the posture-detection models.

Hand Segmentation

With the emergence of new technologies such as virtual reality, researchers are looking for new means to interact in more natural ways. This has led to an emergence of research on detecting body-part and hand information using camera-based technologies to allow users to interact with a computer more naturally without the use of a physical controller. For example, research into hand-pose recognition utilizes depth cameras to identify detailed 3-D information about key features of the hand, such as joint locations in a 3-D space. These joint locations are then used to infer the pose of a hand that can be mapped to a specific action in the user interface. The first step in the process for pose recognition is to segment specific locations of the hand from the depth map. A similar process is needed for posture detection. Before being able to predict hand posture, the pianist's hands must first be segmented from the depth map in which the hands are interacting with a piano.

Body-part segmentation from 3-D data is a well-researched problem in computer vision. One of the original needs was to identify body parts from depth maps to find specific joint locations for body-pose recognition (Shotton et al. 2011). To label 31 parts of the body, per-pixel classification was performed using a random decision forest (RDF) trained with custom depth-image features (DIFs). Similar approaches have been used for hand segmentation in hand-pose recognition. Keskin et al. (2013) used the same approach as as Shotton's team, including the same DIFs, to identify 21 hand parts from a depth map. Thompson and colleagues (2014) used this approach as well, but to segment the entire hand from the depth map rather than individual parts of the hand. Liang, Yuan, and Thalmann (2014) also used per-pixel classification to parse hand parts from a depth image, but they implemented a new feature descriptor, depth-context features (DCFs), for each pixel. The new pixel descriptors improve

segmentation accuracy compared with the DIFs used by Shotton's team. In all of these works, there is a single hand in the scene and the hand is not in physical contact with other objects. In contrast, our work involves a depth-map scene in which two hands are both in physical contact with a piano.

There has also been research into segmenting a hand that is interacting with an object. Liang and colleagues (2016) extended earlier work (Liang, Yuan, and Thalmann 2014) with a system for playing a virtual piano. In that work, fingertips are tracked while tapping on a flat surface to mimic piano playing. To segment the hand they use skin color detection combined with the random sample consensus algorithm for plane fitting to improve the accuracy. Then an RDF is applied to the segmented hand to predict 3-D joint locations. To avoid the added complication of skin detection as well as the challenges of using both color and depth data, we utilize only depth data for hand segmentation. Kang and coworkers (2016) have shown that the per-pixel classification approach can be successfully used to segment a hand interacting with an object using the same DIFs as Shotton. The descriptors were used to train the RDFs using depth maps of participants interacting with various objects.

In this work, we use per-pixel classification using an RDF to segment the left and right hands from the depth-map scene. We experiment training the RDF with both Shotton's DIFs and Liang's DCFs to find the optimal descriptors. The rest of this section discusses, in detail, the process and descriptors used to isolate each hand from a single depth map.

Per-Pixel Classification

The task of per-pixel classification is to predict a category for every pixel in an image or a depth map. For each pixel, features are extracted that are used for training a classification model, such as an RDF. The rest of this section presents the process of training an RDF for per-pixel classification identifying the left hand, the right hand, and the background from a depth map. The classification results of training the RDF with either of two feature descriptors are also presented. First we look at DIFs (Shotton et al. 2011) as used for body part inference, then the more-recent DCFs (Liang, Yuan, and Thalmann 2014).

Depth-Image Features

Depth-image features are discriminative features that compare the depth values of pairs of pixels in a neighborhood to capture a representation of the surrounding context of a given pixel p. Following Shotton et al. (2011), for each feature of p, two offset parameters, u and v, are randomly selected and are used to determine the pixel locations of each offset. The feature is computed as the difference, in depth values, at each offset location. Each feature is computed as

$$f_\theta(I, p) = d_I\left(p + \frac{u}{d_I(p)}\right) - d_I\left(p + \frac{v}{d_I(p)}\right), \qquad (1)$$

where $d_I(p)$ is the depth value of pixel p in image I. To ensure depth invariance, the offsets are normalized to the depth of p using $1/d_I(p)$. A large constant value is given to any offset pixel that lies on the background or outside the bounds of the image.

The offset parameters, u and v, are randomly sampled from a uniform distribution. The range of the offset sampling affects the size of the neighborhood to examine; a small sampling range for the offset values represents a narrow context that is close to the pixel, whereas a large range increases the area being captured by the features.

Figure 4b shows an example of a subset of four randomly sampled pairs of offset locations for a pixel, marked with a black "x" located on the left index finger. Each pair of feature offsets is represented in the figure by a distinct shape. For each offset pair, the difference in depth is calculated using Equation 1. In practice the number of offset pairs is much higher; here we use a small value for the purpose of visualization.

Depth-Context Features

Depth-context features (Liang, Yuan, and Thalmann 2014) provide a more-structured approach to

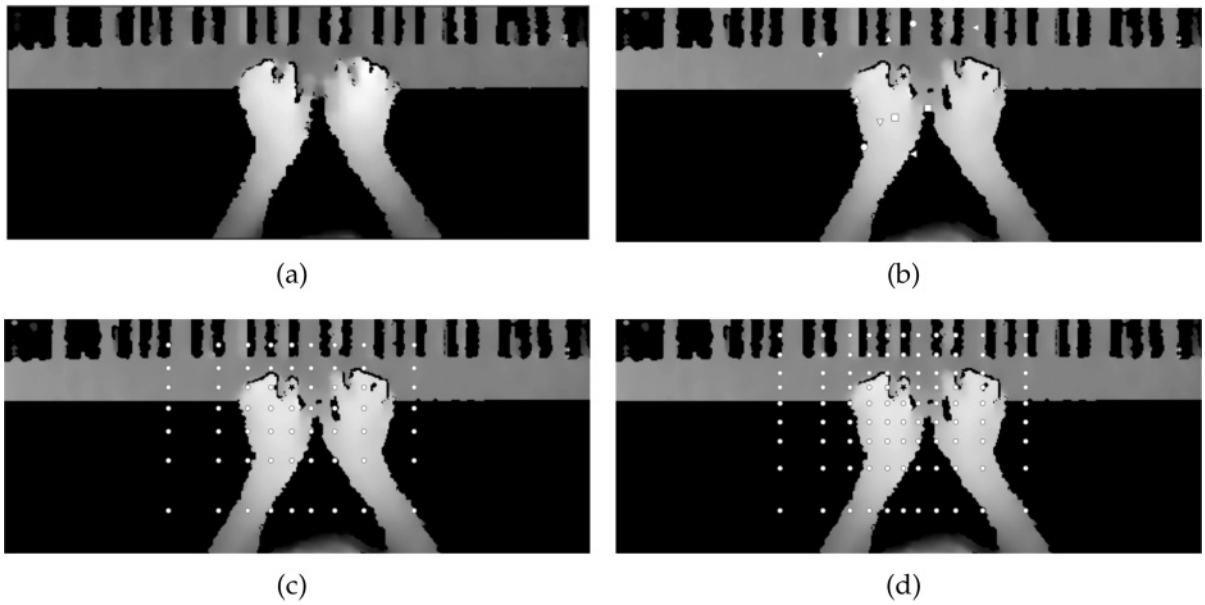


Figure 4. The original depth map (a), and examples of extracting features of a single pixel in a depth map used to classify the pixel as either hand or background of offset: a subset of depth-image feature (DIF) offsets (b), and depth-context feature (DCF) offsets with parameters M = *4 and* r = *.15 (c) and* M = *5 and* r = *.15 (d).*

examining the context of a pixel's neighborhood. Instead of using randomly generated context points, Liang and colleagues assert that points nearer to the classification pixel better describe the context of the pixel as compared with points that are further away. Thus Liang's team propose a distance-adaptive sampling scheme that samples more densely from points closer to the classification pixel. The distance of the context points from the current pixel is defined by maximum range value r, and the parameter M defines the number of context points to sample. Figures 4c and 4d show examples of the selected context pixels using the distance-adaptive approach with different M values for a pixel on the left index finger (marked with a black "x").

Following the method developed by Liang's team, depth invariance is handled using offsets of depth-context points defined in 3-D space rather than the image plane. The location of the 3-D context points relative to pixel p with 3-D coordinates v can be defined as $v_d = [a_d, b_d, 0]^T$. Therefore, to find the pixel coordinates, p_c, we must project the depth context point back to the image plane with $p_c = \Psi_p(v + v_d)$. The feature value for a context point is thus calculated as the difference between the depth of the current pixel and the depth of the context points at the projected pixel coordinate:

$$f_\theta(I, p, v_d) = d_I(p) - d_I[\Psi_p(v + v_d)]], \quad (2)$$

where $d_I(p)$ is the depth at the given pixel as found in the depth map.

Classification Using Random Decision Forests

To predict a category for each depth-map pixel, we utilize an RDF classifier. An RDF is an ensemble classifier composed of T decision trees whose predictions are aggregated using votes weighted by the posterior probabilities to make the final prediction. Each decision tree t is composed of split and leaf nodes. A split node contains a feature and threshold value used to determine the branching direction. A leaf node contains a learned probability distribution $P_t(c|I, x)$ for labels c, where I is the image and x is the pixel to classify.

To train an RDF, a random subsample (sampled with replacement) of the training data is selected to train each tree in the forest. Additional randomness is applied when finding the split parameters of a

node during construction of an individual tree. At each node, a random subset of features is selected for consideration when calculating the criteria for splitting. This approach helps to improve accuracy and reduce overfitting (Breiman 2001). Training samples are generated by randomly sampling N pixels from a depth map for each pixel category and calculating the corresponding feature values for the sampled pixels. This is done for each depth map in the training data to generate a complete training set for the RDF.

To segment the hands from the depth map, each pixel is assigned a label by evaluating all trees in the forest and calculating the weighted average using

$$P(c|I, x) = \frac{1}{T}\sum_{t=1}^{T} P_t(c|I, x). \quad (3)$$

A label l is then assigned to each pixel x of image I by $l = \arg\max_c P(c|I, x)$.

Experiments

To validate the hand segmentation approach previously discussed and to find the optimal feature descriptors, we performed a set of experiments comparing the DCF and DIF descriptors. An RDF classifier, implemented with scikit-learn (Pedregosa et al. 2011) and consisting of ten trees with a maximum depth of 20, was used in the experiments. The rest of this section presents the results of the hand segmentation approach on a real-world data set.

The data set consists of depth maps from the recordings of the students. A subset of the recordings was created by sampling the depth-map recordings every second. This results in a set of 661 depth maps for training. For each of the depth maps, a mask was created to label the pixels as either left hand, right hand, or background.

Results

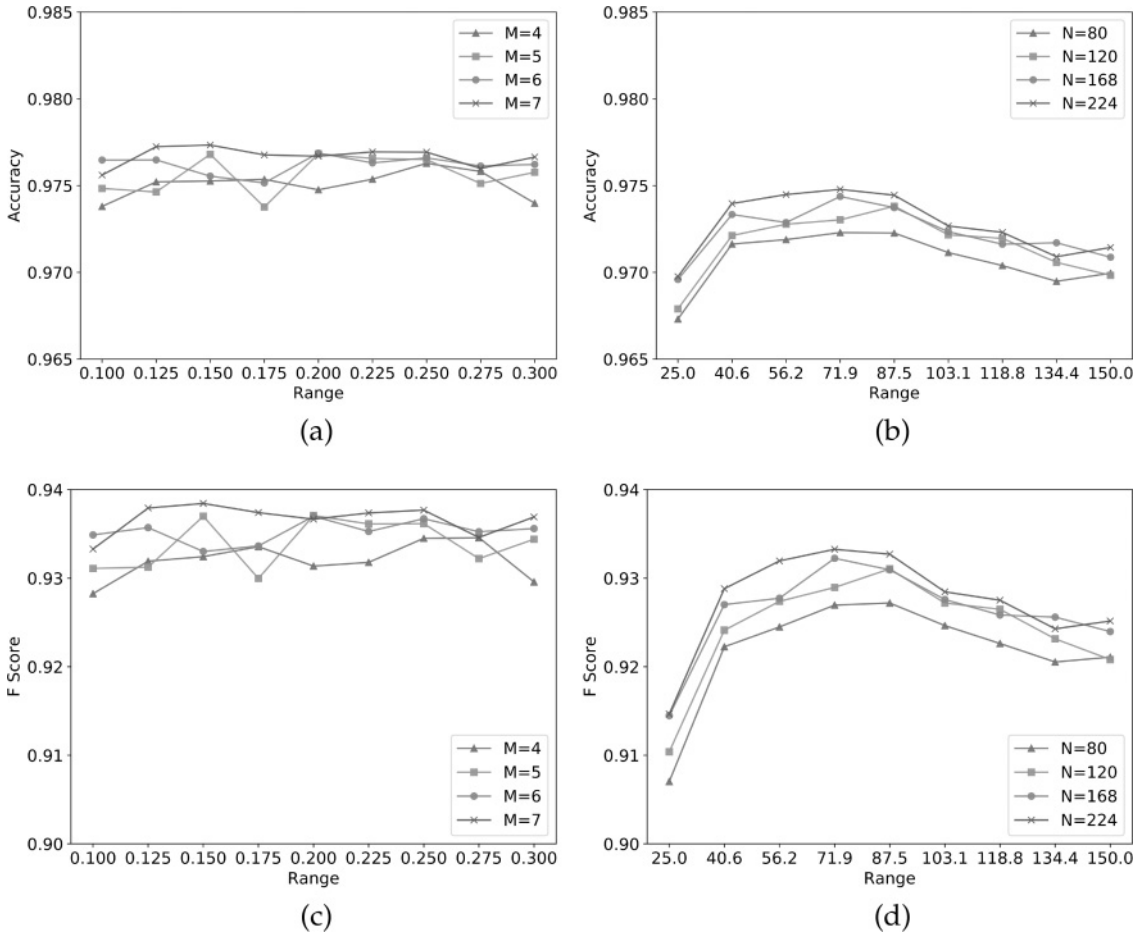
To evaluate our feature sets on the original data set and find optimal parameters, we tested both the DIF and the DCF with varying parameters. Both descriptors have two main parameters to tune. For DCF the parameters are M, which influences the number of features per pixel, and the range for offset selection. DIF has similar parameters: N, the number of features, and the range, indicating the size of the neighborhood for offset selection. The N value for DIF in the experiments corresponds to the number of features for each M of the DCF descriptor. To ensure that the segmentation models are not overfitting to data that has already been seen, we use a participant-based, leave-one-out cross validation. In this scheme we train the segmentation models on all but one participant and use the left-out participant's data for testing the model. The classification accuracy and F-score of each round of cross validation are then averaged. Figure 5 shows the results for each descriptor and parameter set.

Overall, the results show that DCF consistently performs better in terms of both classification accuracy and F-score. Furthermore, we see that generally the more features used in training, the better the performance of the classifier. For real-time prediction, however, we need to select a parameter value that balances accuracy against prediction time. Prediction time is dependent upon the number of features to be extracted. For this reason, we find $M = 5$ and $r = .2$ to be the best combination of DCF parameter values for hand segmentation.

Detecting Hand Posture

With the left and right hands segmented from an input depth map, the next step in the posture-detection pipeline is to extract descriptors from the segmented hand depth maps for use with the posture-detection models. In our previous work (Johnson et al. 2016), two feature descriptors were compared and found to have similar results. Using an expanded real-world data set, we experiment with the same descriptors, histograms of oriented gradients, and histograms of normal vectors. The features are used to train and test a posture-detection model based on support vector machines (SVMs). The rest of this section discusses the feature extraction-process and building the posture-detection model.

Figure 5. Per-pixel classification results of hand segmentation using DCF and DIF with varying range and neighborhood sizes: DCF accuracy (a), DIF accuracy (b), DCF F-score (c), and DIF F-score (d).



Feature Extraction

Two methods for extracting descriptors from the depth maps are compared, histograms of oriented gradients (HOGs, cf. Dalal and Triggs 2005) and histograms of normal vectors (HONVs, see Tang et al. 2013). Histograms of oriented gradients are image descriptors often utilized for object and human recognition with RGB and grayscale images. Although influenced by the HOG approach, the HONV descriptors were specifically designed for depth data, describing the geometry of the surface of objects (Tang et al. 2013).

The key idea behind the HOG approach is to capture local shape through edge strength and direction. In the RGB space, HOG descriptors are calculated by approximating the derivative of color intensity in the x and y directions of an image. The gradients are converted to polar form in order to generate orientation angles and corresponding magnitudes for each pixel in the image. Next, histograms are generated for the image through sliding nonoverlapping windows (or cells). For each cell, orientation angles are voted into bins with the votes weighted by the magnitudes, thus capturing both the direction and strengths of change. Extraction using HOGs also includes a process for normalizing gradient strengths over a block of cells. Dalal and Triggs (2005) explored four different normalization schemes, called L1-norm, L1-sqrt, L2-norm, and L2-Hys. They found that all work equally well except L1-norm, which reduces performance

by 5 percent. For this work, we use L1-sqrt for normalization in all experiments.

Although Dalal and Triggs's work was performed on RGB images, the HOG approach has also been shown to work for object and human detection with depth data (Spinello and Arras 2011; Lai et al. 2011). And although the data is not in the RGB space, the HOG approach calculates the orientation and magnitude of the change in depth values. Thus, when applied to depth maps, these features capture the shape of an object not only via edge direction but also by capturing the depth gradients over the surface of the object. For example, when a pianist is playing with too low a wrist, the gradients of the top of the hand will be greater than when playing in correct form, in which case the top of the hand is flat.

On the other hand, HONV descriptors were developed specifically for depth data to provide a geometric representation of objects (Tang et al. 2013). For HONV descriptors, the x and y gradients are used to calculate the azimuth and zenith angles of normal vectors of unit magnitude. The angles of each pixel in a window are voted into 2-D histograms. The experiments performed by Tang's group showed that HONVs generally perform better than HOGs in object recognition using depth maps. Although we do not apply block normalization to their implementation, we have added L1-sqrt normalization to explore its effects on posture detection.

Training

Owing to wide variations in hand shapes, playing style, and error postures, combined with a limited number of participants for data collection, we decided to use student-specific posture-detection models for training. To validate this approach, we used recordings of depth data from four piano students, each performing the same five beginner piano exercises. Participant 1 (P1) was 12 years old, participant 2 (P2) was 11, participant 3 (P3) was 9, and participant 4 (P4) was 11. The exercises for this study were all in the key of C major and were mostly monophonic, although there were a few simple chords in one of the exercises. The length of each exercise recording varied per student, with an average length of 26 sec. The left and right hands of each frame of the recordings were separately annotated with one of the three posture categories. Using the annotated data, the hands were segmented from the depth map using the procedure outlined above. After segmenting both hands, the right hand was flipped horizontally, giving the image the same orientation as the left hand, affording the ability to train a single detection model for each student (as opposed to individual models for each hand). For each student model, an SVM was then trained using the features extracted from both hands. The following section discusses the results of training with each descriptor and the effects of various parameter values.

Experiments

Both feature descriptors have a number of parameters to tune for optimal prediction. In this section, we present the results of experiments for parameter tuning to find the best descriptor and parameter values for posture detection. Due to the high dimensionality of the data, SVMs are used for prediction. The rest of this section explores the effects of employing the two descriptors and their corresponding parameters for posture detection. To validate the training, we use threefold cross validation using 1-second windows of the recordings. This scheme is meant to reduce the overfitting effects seen with standard cross validation of sequential images, in which neighboring frames, which have minimal variation, are split into the training and testing data.

Image Size

In general image-processing algorithms, such as those for object recognition, the input images are required to be rescaled to a constant size, such as 128×128, for a consistent size of feature vectors. As posture detection requires information that is more fine-grained than does general object recognition, rescaling could lead to information loss that would

Table 1. Average Accuracy of Hand-Posture Detection

	128 × 128	*90 × 160*	*130 × 190*	*96 × 96*
HOG	93.3%	92.9%	93.3%	93.1%
HONV	94.7%	94.8%	94.6%	94.8%

Hand posture detected using histogram of oriented gradients (HOG) and histogram of oriented normal vectors (HONV) descriptors.

affect detection performance. Using larger images, however, leads to larger feature vectors that may affect runtime performance. Furthermore, reducing the image to a square changes the aspect ratio of the extracted hand regions, which was found to have an average ratio of 9:16. Rescaling to this ratio may represent the shape of the hand more accurately but can also lead to information loss. The largest hand region was found to be 130×190; so to keep all the hand information available we also experimented with increasing all images to this size by padding the front of each image axis with zeros in order to keep a consistent size without rescaling the image. For this experiment we used a default cell size of 8×8 and the default block size of 3×3 and an SVM with a linear kernel.

As shown in Table 1, the various image sizes appear to have limited effect on prediction accuracy, with HONVs slightly outperforming HOGs in all cases. Figure 6 shows the prediction results of the image sizes for each participant, which paints a slightly different picture. Although the image sizes have limited effect on accuracy, the HONV approach improves the accuracy for the hardest case participant: P3 benefits from an average 4.5 percent increase in accuracy using HONV descriptors. As image size has negligible effects on performance, we used a scaled image size of 128×128 for the rest of the experiments.

Cell and Block Sizes

Dalal and Triggs (2005) found the optimal cell size for detecting humans in images with HOG descriptors to be 6×6 and the optimal block size to be 3×3. Because posture detection benefits from an understanding of the full geometric shape of the hand, not just edge shape, different cell and block sizes may have different results for posture detection. Figure 7 presents the results of testing a range of cell and block sizes for each descriptor. The HONV approach was also tested without block normalization. In the case of posture detection, a smaller cell size results in increase prediction performance. (Due to computational resources we omit a cell size of 4×4 for the HONV approach. Because this approach use 2-D histograms, the resulting feature vectors require significantly more space than the HOG approach.) Similar to the results of Dalal and Triggs, block sizes of 2×2 and 3×3 tend to work best for hand-posture detection using HOGs. On the other hand, HONVs usually benefit from normalization, but they are less affected by block size.

Exercise-Based Training and Oversampling

In the previous experiments, cross validation was performed by partitioning the data into 1-sec windows and splitting the windows into training or testing sets. In this section, we utilize a "leave-one-exercise-out" cross validation approach to validate models of posture detection that were trained for individual participants. In this scheme, cross validation is performed by training the model with four of the five exercises, and the model is tested on the exercise that was left out. For this experiment, prediction is performed by training an SVM using an RBF kernel with $C = 10$ and $\gamma = .01$, using the HONV descriptor with 8×8 pixels per cell and 1×1 blocks for normalization.

One of the challenges of using customized prediction models is that there is little control over the number of samples collected per category, potentially leading to an unbalanced data set. Table 2 provides an overview of the category counts per participant, indicating that each participant is prone to different distributions of posture categories, with some categories having relatively few samples.

Two common methods for dealing with unbalanced data are majority undersampling and minority oversampling. Undersampling is not a good idea in this case, because it would require the data to be

Figure 6. Individual participant posture detection accuracy of different depth map sizes using: HOG (a) and HONV (b) descriptors.

Figure 7. Accuracy of hand-posture detection for HOG (a) and HONV (b) descriptors with different cell and block sizes.

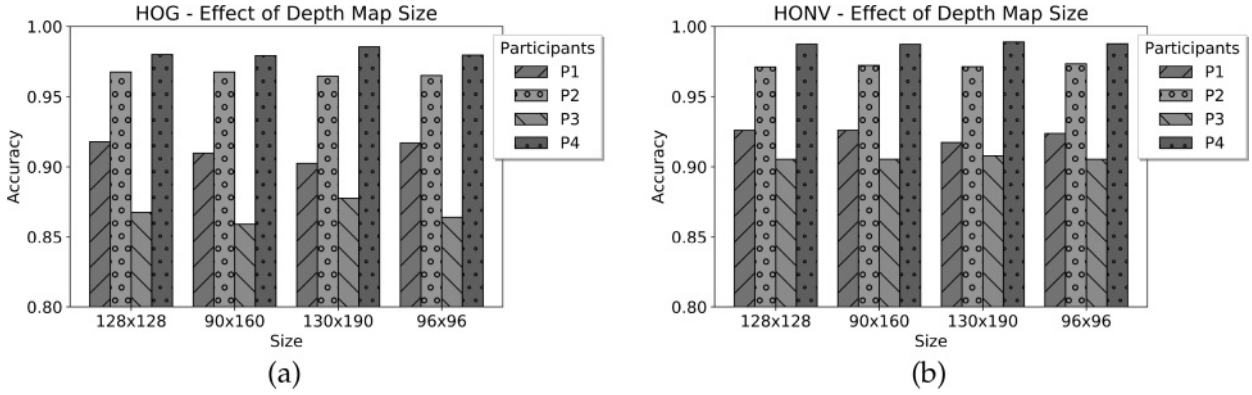


Figure 6

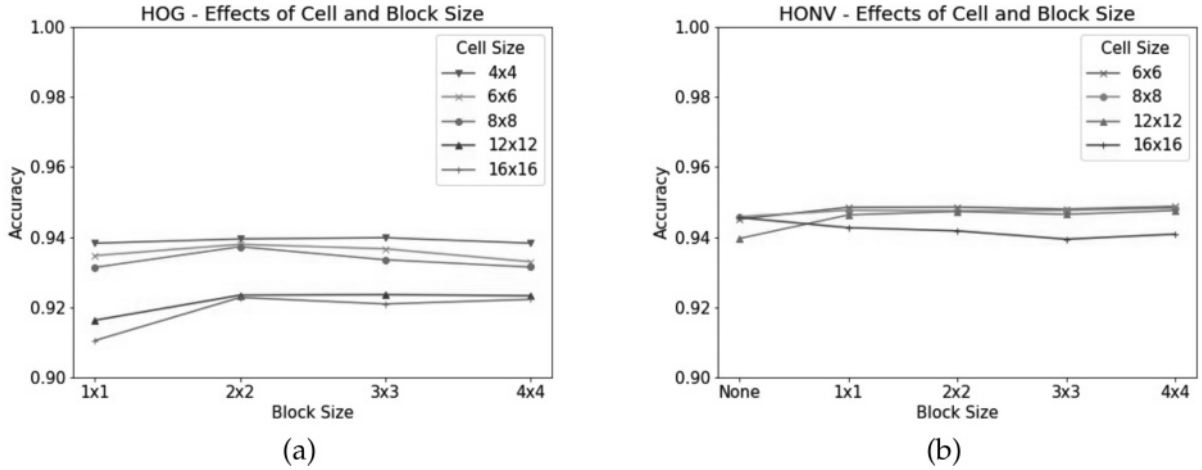


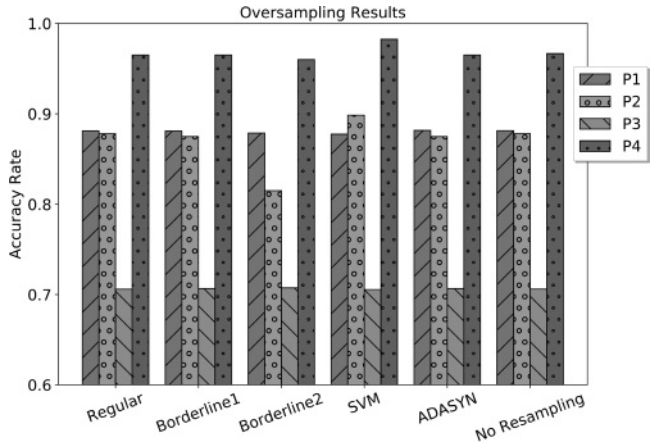
Figure 7

Table 2. Hand-Posture Category Counts for Test Participants (P1–P4)

	Correct	*Low Wrists*	*Flat Hands*
P1	6,011	162	1,021
P2	3,917	1,336	47
P3	2,262	3,376	0
P4	6,111	286	27

under-sampled to the size of the smallest class. In this case, our data would not be large enough to train a robust model. Instead, we use oversampling to balance the data, testing both the synthetic minority oversampling technique (SMOTE) and adaptive synthetic sampling (ADASYN). Rather than simply oversampling with replacement, SMOTE oversamples by generating data in the feature space by calculating features for synthetic samples that lie between a minority sample, x_i, and a neighbor, x_{zi}, selected randomly from the k nearest neighbors. The new features are calculated using $x_{new} = x_i + \lambda(x_{zi} - x_i)$ where λ is a value between between 0 and 1 selected randomly for each sample (Chawla et al. 2002). Similarly to SMOTE, ADASYN (He et al. 2008) uses interpolation to generate new

Figure 8. Accuracy of hand-posture detection using different oversampling methods for balancing the training data of Participant (P) 1–4.



samples, but it is biased to select samples that are harder to learn. In other words, more synthetic samples are generated for samples that are hard to learn, effectively adapting the decision boundary towards the hard-to-learn samples. Oversampling, with SMOTE or ADASYN, generates a balanced data set for training the posture-detection models.

There are four SMOTE variations for selecting minority samples to use for sample generation. Regular SMOTE simply uses a random selection from all possible minority samples (Chawla et al. 2002). The Borderline-1 and Borderline-2 SMOTE variations classify minority samples as "in danger" if fewer than half the neighboring samples are from the same class. The in-danger samples are then selected to use for new sample generation (Han, Wang, and Mao 2005). The fourth variation, SVM SMOTE, takes the support vectors of a trained SVM into consideration to select the samples used for new sample generation (Nguyen, Cooper, and Kamei 2011). Figure 8 shows the results of the prediction models for each of the individual participants, using either of the SMOTE variants or ADASYN to balance the data sets. Most of the oversampling variants have little effect on performance of the inference models, but there are a few exceptions. The SVM SMOTE variant shows improved accuracy for participants P2 and P4, and Borderline-2 shows a decrease in accuracy for P2. Little change is shown for P3 with each technique, because the data were already well balanced between two classes. Furthermore, a review of the participant-based confusion matrices from models trained with the SVM SMOTE (see Figure 9b) compared with the confusion matrices for models trained with no oversampling (see Figure 9a) shows that SVM SMOTE improves prediction for certain minority classes. For example, there are improvements in the "flat hands" class for P1, as well as the "low wrists" class for P4. In cases where the number of samples is substantially smaller than the majority class, oversampling does not provide an improvement.

In this section we have detailed the process and results of tuning individual posture-detection models and the features used to train the models. Considering a trade-off between accuracy and runtime performance, it was found that HONV descriptors with a cell size of 8×8 and a block size of 1×1 are optimal for a depth map of 128×128 pixels. The results of the exercise-based scheme for cross validation show that it is possible to achieve a working prediction model using as little as four exercises for training. This is, however, dependent upon the severity and frequency of a student's errors in posture. Such considerations would need to be made when designing a detection-training interface. In cases where the third posture category is too small, the model could instead be trained as a binary classifier until enough samples of the category are recorded.

Discussion

This work presented and evaluated an approach for the automatic assessment of pianist hand posture using data recorded with a depth camera. Implementing this system into a CAMIT interface requires converting assessment output (i.e., the detected hand-posture class) into feedback that is presented to the student. Further, utilizing student-specific detection models raises a number of implications for system design. In the rest of this section we discuss considerations for assessment feedback as well as some possible solutions to the implications of training student specific models.

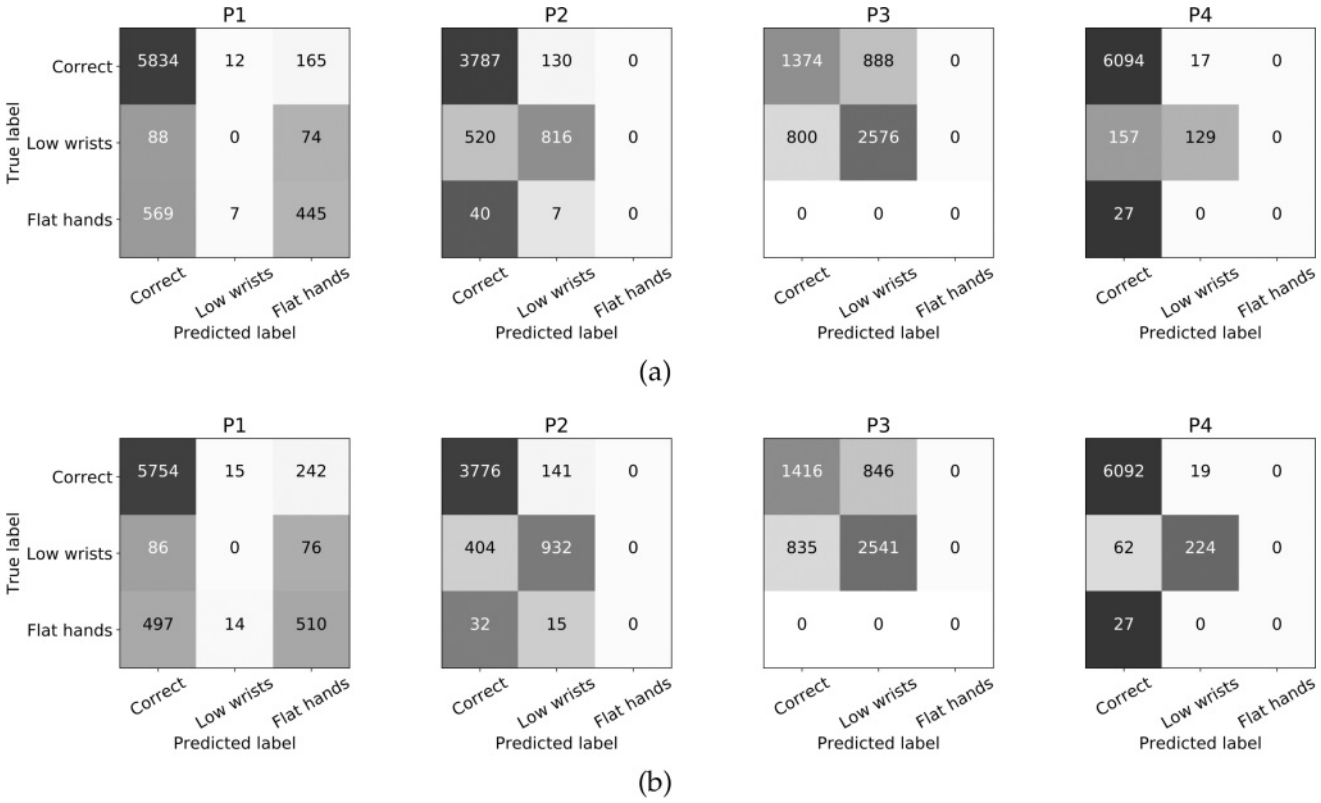


Figure 9. Confusion matrices for each student posture model trained without oversampling (a) and with a support vector machine–based synthetic minority oversampling technique, SVM SMOTE (b).

Considerations for Interface Design

Based on our review of the CAMIT literature, we find three main techniques that may be used to provide beginning students with feedback about their performance: (1) real-time feedback with auditory cues (Ferguson 2006; Ng et al. 2007), (2) video playback of a practice session augmented with visual feedback (Ng et al. 2007), and (3) performance-quality scores and visualizations (Blanco and Ramirez 2019). To be effective, the interface must be motivating and informative, and it must help the student improve. As previous research has shown these techniques to be effective, an ideal interface may provide elements of each method. Choosing the proper feedback method, however, must take a few aspects into consideration: the amount of information presented to the student, the student's ability to understand and process the information, and the robustness of the detection system for providing correct information.

The simplest feedback method would be to provide students with a single score or visualization to indicate the quality of their performance. To assist with self-evaluation, this method would allow students to compare their performance during a practice session to their performance of a previous session or to the performance of an expert. Furthermore, this design would allow a teacher to quickly track students' progress through sessions in which the teacher is not present. One of the benefits of such a system is that it would be the easiest for a student to understand, making it ideal for young or beginning students. Additionally, using a score-based method would support a tutoring system with gamification to motivate students. Technically, the scoring method would be the easiest to implement because it is the least vulnerable to posture classification

errors, as improvement is relative to past performances and minor errors in classification would not be noticed as explicitly. The main drawback is a lack of context to indicate what mistakes were made and when they were made. Without the detailed information students may not know exactly how to improve their performance, especially if a teacher is not available. Previous research into a visual feedback system for performance quality, however, shows this method to be effective for improving performance (Blanco and Ramirez 2019).

A more informative approach to presenting performance feedback would be video playback of the performance, augmented with visual indication of posture errors. With this method, students are able to view exactly when and how mistakes were made. Furthermore, as opposed to real-time feedback, students are able to analyze their performance while not focused on the other cognitively demanding aspects of practice, such as playing the correct notes. There are some challenges to using such a system, though. Namely, the detection accuracy must be near perfect, as detection errors may adversely affect a student's ability to self-evaluate. Furthermore, students (especially young ones) may find watching a recording of their performance to be boring, demotivating, or both.

Providing real-time feedback, instead, may address motivation issues by integrating feedback directly into the practice session. Providing real-time auditory cues immediately when mistakes are recognized is already familiar to students, since this is similar to the style of feedback they would receive during training sessions with a teacher. A system for beginners should only alert the student to an issue after a specific period of time playing with poor technique, as continuous feedback may be too cognitively demanding. With this method, once the detection system recognizes that a student performed with incorrect posture for a number of seconds, it could trigger an auditory alert, such as "remember to keep your wrists up." This method would help students self-evaluate by receiving auditory cues exactly at the moment they occur, allowing students to quickly adjust their technique. Although the real-time feedback may be cognitively challenging, it is most similar to the feedback they already receive from their teacher.

As technology in mixed reality (MR) advances, implementing immersive systems may also prove to be effective in music tutoring by providing real-time feedback. For example, our system for hand-posture detection could be implemented with an MR device such as the Microsoft HoloLens. With this interface, students would be presented with computer-generated visual cues overlaid upon their hands, with feedback directing them on how to adjust their hands for correct posture. We have started exploring real-time feedback through MR for music tutoring (Johnson and Tzanetakis 2017), but it has yet to be seen how effective this method is for music tutoring. Future research is still needed to provide guidance on how best to design MR interfaces for real-time hand-posture correction.

Future Work

This study lays the groundwork for an automatic assessment of hand posture to enhance piano pedagogy for beginning piano students, but there are still two main challenges to address. First is accuracy of the information provided by the detection system and the robustness to variations in hand formation not related to posture. The data used in the experiments were taken from typical exercises for beginning students, so there is only minimal variation in hand movement and deformations, such as the lateral spread of the fingers. Thus, the detection system in its current form may not be robust enough to scale to more-advanced techniques required of students as they improve. Second, using a per-user training scheme requires effort from the teacher and the student to train the model before use. If too much effort is required for training, the system becomes impractical. We leave these challenges for future work but discuss possible methods for addressing them here.

One potential solution is to build a larger data set with greater variation of hand shapes and playing styles, with the goal of improving the generalization of the detection model. One of the biggest challenges with machine learning, however, is that building

generalized models requires large-scale data sets; for example, one of the largest data sets used in machine learning research, ImageNet (Deng et al. 2018), has has over 14 million images at the time of writing (see http://image-net.org/about-stats). This especially becomes a challenge when working on new problems, such as the detection of a pianist's hand posture, that have little or no existing data and whose data require domain experts such as piano teachers for annotation. Furthermore, students' hand-posture errors may not be limited to those we present in this article. Although this could be addressed through one-class classification, in which training is performed using only correct posture, such a system would not be able to provide a student with information about how to correct errors. To address the challenges related to large-scale data collection, we propose a per-user training system for posture detection in which student and teacher work together with the interface to train the posture-detection system. We have shown with our research that this is possible with limited amounts of data.

A per-user training scheme has the benefit that detection models can be customized to each student's skill level and can overcome the challenges in obtaining enough data for generalization. Customization may be achieved by allowing teachers to define their own posture categories and to choose the appropriate training exercises that match the students' skill levels and playing style. Giraldo et al. (2019) took a similar approach in their work on prediction of tone quality to overcome challenges of subjectivity in tone perception. Per-user training is not without its drawbacks, however. Most notable is the fact that it takes time and effort from both the teacher and the student to train the models. If training is too arduous, such as labeling an entire recording, then the system will not be used. Additionally, teachers cannot be expected to be experts in machine learning, so a training system should be easy to understand. To address difficulties such as these, there is emerging work in "human-in-the-loop" machine learning, such interactive machine learning (Amershi et al. 2014; Holzinger 2016; Chen et al. 2018) and active learning (Settles 2009), in which humans work directly with a training system to build and improve learning models. Active learning works by selecting samples to be labeled based on some criterion, such as maximum uncertainty, then asking a human participant to label the selected samples. Interactive machine learning (IML) builds on this idea with a focus on designing interfaces in which humans work with the machine to improve the learned model through iterative train-feedback-correct cycles (Amershi et al. 2014). Integrating IML and active learning techniques into model training will help improve model robustness by making per-user training feasible and will help improve accuracy through iterative training cycles.

Assuming we are able to achieve near-perfect classification of hand posture with the methods proposed above, there are still times in which it may not be appropriate to analyze a student's hand posture, such as when the hand is in transition. Although transitions are generally minimal for beginning students, providing posture feedback for more-advanced students should ignore these transient periods, so as not to provide incorrect feedback. It may be possible to address this by adding another category to the posture-detection model to identify hand positions that should be ignored. A more robust method may be to integrate gesture detection that tracks hand motion, to first identify when hands are in an appropriate state for posture detection. Integrating these capabilities would improve the decision about when to present posture feedback to the student.

Conclusion

In this article we have presented a system for detecting piano stutents' mistakes in hand posture from a single frame of a depth-camera recording. Using a per-pixel classification scheme for hand segmentation, we found that the DCF descriptors developed by Liang, Yuan, and Thalmann result in the best segmentation accuracy. The higher accuracy, compared with DIF, is due to DCF's implementing denser sampling of context offsets closer to the pixel being classified. Although we achieve positive results with the current sampling distribution, there are still misclassified pixels in areas where the hand is in direct contact with the

piano. To account for this, future work using a sampling distribution that is even more densely sampled near the classification pixel should provide context that is more fine-grained, providing better classification. With hands segmented from the image, an SMV is used to detect the posture of each hand. Evaluation of posture-detection models showed that the HONV descriptors developed by Tang and coworkers provide the best performance. Further, the HONV approach was improved by adding block normalization to the process of feature extraction. To account for shape and size variations in hands as well as varying practice environments, we implement detection models customized and trained for individual students. Using individual models presents a problem, however, as students will not always perform with an equal distribution of posture categories, resulting in unbalanced data for training. Our experiments have shown that this problem can be addressed with oversampling in the feature space using SMOTE. The results presented in this work show the effectiveness of the proposed computer-vision pipeline for posture-detection models trained for individual piano students. We have discussed thoughts on designing interfaces to provide feedback to piano students using the detection system, as well as on designing interfaces using IML to improve the process of training individually customized detection models. The design of these interfaces is left for future research.

References

Amershi, S., et al. 2014. "Power to the People: The Role of Humans in Interactive Machine Learning." *AI Magazine* 35(4):105–120.

Blanco, A. D., and R. Ramirez. 2019. "Evaluation of a Sound Quality Visual Feedback System for Bow Learning Technique in Violin Beginners: An EEG Study." *Frontiers in Psychology* 10:Art. 165.

Breiman, L. 2001. "Random Forests." *Machine Learning* 45(1):5–32.

Burnam, E. M. 2005. *A Dozen a Day Preparatory Book*. Milwaukee, WI: Willis.

Carfagni, M., et al. 2017. "On the Performance of the Intel SR300 Depth Camera: Metrological and Critical Characterization." *IEEE Sensors Journal* 17(14):4508–4519.

Chawla, N. V., et al. 2002. "SMOTE: Synthetic Minority Over-Sampling Technique." *Journal of Artificial Intelligence Research* 16(1):321–357.

Chen, N.-C., et al. 2018. "AnchorViz: Facilitating Classifier Error Discovery through Interactive Semantic Data Exploration." In *Proceedings of the International Conference on Intelligent User Interfaces*, pp. 269–280.

Dalal, N., and B. Triggs. 2005. "Histograms of Oriented Gradients for Human Detection." In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 886–893.

Dalmazzo, D. C., and R. Ramirez. 2019. "Bowing Gestures Classification in Violin Performance: A Machine Learning Approach." *Frontiers in Psychology* 10:Art. 344.

Dannenberg, R. B., et al. 1990. "A Computer-Based Multi-Media Tutor for Beginning Piano Students." *Interface* 19(2–3):155–173.

Dannenberg, R. B., et al. 1993. "Results from the Piano Tutor Project." In *Proceedings of the Biennial Arts and Technology Symposium*, pp. 143–150.

Deng, J., et al. 2018. "ImageNet: A Large-Scale Hierarchical Image Database." In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 248–255.

Ferguson, S. 2006. "Learning Musical Instrument Skills through Interactive Sonification." In *Proceedings of the International Conference on New Interfaces for Musical Expression*, pp. 384–389.

Giraldo, S., et al. 2019. "Automatic Assessment of Tone Quality in Violin Music Performance." *Frontiers in Psychology* 10:334.

Hadjakos, A. 2012. "Pianist Motion Capture with the Kinect Depth Camera." In *Proceedings of the Sound and Music Computing Conference*, pp. 303–310.

Hadjakos, A., F. Lefebvre-Albaret, and I. Toulouse. 2009. "Three Methods for Pianist Hand Assignment." In *Proceedings of the Sound and Music Computing Conference*, pp. 321–326.

Han, H., W.-Y. Wang, and B.-H. Mao. 2005. "Borderline-SMOTE: A New Over-Sampling Method in Imbalanced Data Sets Learning." In D.-S. Huang, X.-P. Zhang, and G.-B. Huang, eds. *Advances in Intelligent Computing: International Conference on Intelligent Computing*. Berlin: Springer, pp. 878–887.

He, H., et al. 2008. "ADASYN: Adaptive Synthetic Sampling Approach for Imbalanced Learning." In *Proceedings of the IEEE International Joint Conference on Neural Networks*, pp. 1322–1328.

Holzinger, A. 2016. "Interactive Machine Learning for Health Informatics: When Do We Need the Human-in-the-Loop?" *Brain Informatics* 3(2):119–131.
Johnson, D., and G. Tzanetakis. 2017. "VRMin: Using Mixed Reality to Augment the Theremin for Musical Tutoring." In *Proceedings of the Conference on New Interfaces for Musical Expression*, pp. 151–156.
Johnson, D., et al. 2016. "Detecting Pianist Hand Posture Mistakes for Virtual Piano Tutoring." In *Proceedings of the International Computer Music Conference*, pp. 168–171.
Kang, B., et al. 2016. "Hand Segmentation for Hand–Object Interaction from Depth Map." In *Proceedings of the IEEE Global Conference on Signal and Information Processing*, pp. 259–263.
Keskin, C., et al. 2013. "Real Time Hand Pose Estimation Using Depth Sensors." In A. Fossati et al., eds. *Consumer Depth Cameras for Computer Vision: Research Topics and Applications*. Berlin: Springer, pp. 119–137.
Lai, K., et al. 2011. "A Large-Scale Hierarchical Multi-View RGB-D Object Dataset." In *Proceedings of the IEEE International Conference on Robotics and Automation*, pp. 1817–1824.
Li, M., et al. 2014. "Using the Kinect to Detect Potentially Harmful Hand Postures in Pianists." In *Proceedings of the Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, pp. 762–765.
Liang, H., J. Yuan, and D. Thalmann. 2014. "Parsing the Hand in Depth Images." *IEEE Transactions on Multimedia* 16(5):1241–1253.
Liang, H., et al. 2016. "Barehanded Music: Real-Time Hand Interaction for Virtual Piano." In *Proceedings of the ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games*, pp. 87–94.
Lu, H., et al. 2008. "iDVT: An Interactive Digital Violin Tutoring System Based on Audio-Visual Fusion." In *Proceedings of the ACM International Conference on Multimedia*, pp. 1005–1006.
MacRitchie, J., and A. P. McPherson. 2015. "Integrating Optical Finger Motion Tracking with Surface Touch Events." *Frontiers in Psychology* 6:Art. 702.
Mora, J., et al. 2006. "Assisted Piano Pedagogy through 3D Visualization of Piano Playing." In *Proceedings of the IEEE International Workshop on Haptic Audio Visual Environments and Their Applications*, pp. 157–160.
Ng, K., P. Nesi, and V. S. Marta. 2008. "i-Maestro: Technology-Enhanced Learning and Teaching for Music." In *Proceedings of the International Conference on New Interfaces for Musical Expression*, pp. 225–228.
Ng, K. C., et al. 2007. "3D Augmented Mirror: A Multimodal Interface for String Instrument Learning and Teaching with Gesture Support." In *Proceedings of the International Conference on Multimodal Interfaces*, pp. 339–345.
Nguyen, H. M., E. W. Cooper, and K. Kamei. 2011. "Borderline Over-Sampling for Imbalanced Data Classification." *International Journal of Knowledge Engineering and Soft Data Paradigms* 3(1):4–21.
Oka, A., and M. Hashimoto. 2013. "Marker-Less Piano Fingering Recognition Using Sequential Depth Images." In *Proceedings of the Korea-Japan Joint Workshop on Frontiers of Computer Vision*, pp. 1–4.
Pedregosa, F., et al. 2011. "Scikit-Learn: Machine Learning in Python." *Journal of Machine Learning Research* 12:2825–2830.
Percival, G., Y. Wang, and G. Tzanetakis. 2007. "Effective Use of Multimedia for Computer-Assisted Musical Instrument Tutoring." In *Proceedings of the International Workshop on Educational Multimedia and Multimedia Education*, pp. 67–76.
Raptis, S., et al. 2005. "IMUTUS: An Effective Practicing Environment for Music Tuition." In *Proceedings of the International Computer Music Conference*, pp. 383–386.
Riley, K., E. E. Coons, and D. Marcarian. 2005. "The Use of Multimodal Feedback in Retraining Complex Technical Skills of Piano Performance." *Medical Problems of Performing Artists* 20(2):82–88.
Salgian, A., and D. Vickerman. 2016. "Computer-Based Tutoring for Conducting Students." In *Proceedings of the International Computer Music Conference*, pp. 159–162.
Schoonderwaldt, E., A. Askenfelt, and K. F. Hansen. 2005. "Design and Implementation of Automatic Evaluation of Recorder Performance in IMUTUS." In *Proceedings of the International Computer Music Conference*, pp. 97–103.
Settles, B. 2009. "Active Learning Literature Survey." Technical Report 1648. University of Wisconsin-Madison, Department of Computer Science.
Shotton, J., et al. 2011. "Real-Time Human Pose Recognition in Parts from Single Depth Images." In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1297–1304.
Spinello, L., and K. O. Arras. 2011. "People Detection in RGB-D Data." In *Proceedings of the International Conference on Intelligent Robots and Systems*, pp. 3838–3843.

Tang, S., et al. 2013. "Histogram of Oriented Normal Vectors for Object Recognition with a Depth Sensor." In K. M. Lee et al., eds. *Computer Vision: ACCV 2012*. Berlin: Springer, pp. 525–538.
Tits, M., et al. 2015. "Feature Extraction and Expertise Analysis of Pianists' Motion-Captured Finger Gestures." In *Proceedings of the International Computer Music Conference*, pp. 102–105.
Tompson, J., et al. 2014. "Real-Time Continuous Pose Recovery of Human Hands Using Convolutional Networks." *ACM Transactions on Graphics* 33(5): Art. 169.