Key Detection for a Virtual Piano Teacher

Adam Goodwin, Richard Green
Department of Computer Science and Software Engineering
University of Canterbury
Christchurch, New Zealand
adg59@uclive.ac.nz, richard.green@canterbury.ac.nz

Abstract—**We propose a method for identifying a piano keyboard present in the video footage of a standard webcam with the goal of teaching chords, scales and suggested finger positions to a beginner pianist. Our keyboard identification method makes use of binary thresholding, Sobel operators and Hough transforms, as well as proposed algorithms specific to this application, to first find an area resembling a piano keyboard before narrowing the search to detect individual keys. Through the use of our method the keys of a piano keyboard were successfully identified from webcam video footage, with a tolerance to camera movement and occluded keys demonstrated. This result allowed the augmented reality style highlighting of individual keys, and the display of suggested fingering, for various chords and scales – which demonstrates the potential for our piano teacher program as a learning tool. The demo application achieved an average frame rate of 25.1 frames per second when run on a 2.20GHz dual-core laptop with 4GB RAM; a suitable rate for real-time use.**

Keywords-segmentation; augmented reality; education; piano keyboard

I. INTRODUCTION

In this paper we present a webcam-based "virtual piano teacher" application. As part of the solution, we propose a method for identifying a piano keyboard, and its individual keys, in video sourced from a typical webcam.

Piano playing requires learning a variety of both theoretical and practical skills. One of the first practical skills a pianist will learn is the ability to play simple chords and scales. An important part of this is choosing suitable fingers to play each note, as this will affect a player's fluidity and efficiency – the importance of which becomes greater and greater as the player advances to playing more difficult pieces. It is the learning of these fundamental skills that we aim to accelerate with our virtual piano teacher.

The overall vision for the virtual piano teacher is for it to allow a user to learn by watching their hands indirectly on a computer monitor. For this to be possible, our application must be real-time. By mounting a webcam above their piano, facing down towards the keys and hands, the player's computer can display in one place everything that the player needs to see. The intent is that highlights and fingering information can be overlaid on the webcam feed depending on the chord or scale being practiced. To achieve this, our keyboard detection method is required to distinguish between the keys of the piano. This approach, as opposed to traditional methods such as taking piano lessons, or learning from books or the internet, allows the player to see exactly what they need to do simply by looking at their hands through the intermediate layer of the computer screen.

II. RELATED WORK

There are many previous works relevant to this paper, including several with a specific focus on piano tutoring with computer vision [1-5]. Furthermore, there is a significant amount of existing work with an emphasis on finger detection and/or piano playing technique [1, 2, 5-11]; however there are relatively few papers which look at keyboard detection in detail [3, 4, 12] – as is the primary technical focus of this paper.

We began by examining the recent papers from our own institution, the University of Canterbury, where work, similar to that proposed in this paper, has been developed in the past [1, 2, 8]. For example, Lang produced a webcam-based piano application – however the focus of this was on entertainment, not education, and there was no attempt made to detect the keyboard or keys of a real piano [8].

Deaker proposed a piano tutoring system which would test the user's knowledge by asking them to play particular notes or chords on a paper keyboard with a printed design [2]. Deaker's proposal is similar to ours in that it requires the detection of the printed keyboard's individual keys in order to determine if the user has followed instructions correctly. However, Deaker's design has several limitations, perhaps the most significant of which being that the identification of keys is not entirely achieved by computer vision alone. A particular number of keys are required to be visible to the camera during calibration (14 white keys, starting from a C key and ending on the B key just under two octaves above) and the software assumes that this requirement has been met.

Another computer vision based piano tutor was proposed by Dawe [1]. In this case, the system is intended for relaying a student's choice of fingering to their teacher during a remote piano lesson. Similarly to Deaker's proposal, Dawe's method requires a calibration stage where the user is required to select a region of interest – in order to narrow the video frames down to the area occupied by the instrument's keyboard. This brings with it the same limitations imposed by Deaker's method; that is the camera must remain stationary at all times, and the keyboard must not be obscured during the calibration process.

978-1-4799-0883-7/13/$31.00 ©2013 IEEE

To find the key locations during the calibration phase, Dawe used horizontal and vertical edge detection to determine the boundaries of the keyboard and its individual keys. Unfortunately, a major limitation of Dawe's method was that a translation between these detected edges and a description of the location of each key of the keyboard was unable to be implemented.

After examining these proposed methods it was clear that a satisfactory method of keyboard and key detection had not been found. The following works were more successful in their attempts to detect individual piano keys.

The work by Barakonyi and Schmalstieg demonstrates a variety of 3D augmented reality (AR) applications, including the *Augmented Piano Tutor* application [12]. The application displays an AR overlay on the user's electronic keyboard, indicating which keys are to be played.

Unfortunately, the paper by Barakonyi and Schmalstieg does not describe their implementation details – making it difficult to learn any lessons from their AR solution. However, some of their high level implementation is explained, and even from this it is possible to suggest some improvements – albeit minor ones. The Augmented Piano Tutor application makes use of ARToolKit for 3D registration [13]. The use of ARToolKit means that a fiducial marker is required as a reference point for the application. It would be most convenient for the user if they did not have to correctly place a marker, or do any more than set up their webcam and laptop, to use a tutoring program. An ideal system would allow the user to simply run an application, point their webcam at a keyboard, and begin playing.

The method proposed by Gorodnichy and Yogeswaran attempts to solve the same problem as identified by Dawe; that is, Gorodnichy and Yogeswaran present a solution to allow current remote piano teaching systems to communicate fingering information [4]. There is one major limitation of Gorodnichy and Yogeswaran's method; the camera used is not an off-the-shelf webcam, but instead a more expensive video camera with high quality output and zoom functionality is required.

The final keyboard detection method we shall discuss is from a paper by Huang, Zhou, Yu, Wang and Du [3], where a markerless AR piano system is proposed. The 3D AR result is very similar to that achieved by Barakonyi and Schmalstieg, although in this case a fiducial marker is not required. To identify the keys of the keyboard, a similar method to that of Gorodnichy and Yogeswaran is employed. After identifying the piano keys, this solution uses OpenGL to augment the video stream with 3D graphics.

This was a paper that seemed to accomplish almost all of what we were setting out to achieve. Consequently, the work by Huang, Zhou, Yu, Wang and Du is regarded as a target result for our paper, with our application intended to match the results produced by the markerless AR piano system as best as possible. More specifically, the system by Huang et al. managed the keyboard identification and rendering of the video feed, with overlays, at a frame rate of approximately 15 frames per second. We hoped to match or exceed this performance with our solution.

As a final note on the system developed by Huang et al., there is a small way in which the design could arguably be improved upon: The markerless AR piano system requires a one-time calibration for each camera and/or piano used, and for the 3D registration to be possible, the dimensions of the keyboard need to be known by the software. This requires that the user manually measure their piano keyboard, then input the information into the AR program. At the expense of 3D augmentations, this potential inconvenience can be removed.

III. SOLUTION

In this section we describe our solution for implementing the proposed virtual piano teacher application. In addition to our overall goal of teaching through augmented reality, with highlighted keys and fingering information, the approaches taken in related works exposed several ways in which a new piano tutoring system can improve on existing alternatives. These potential improvements created additional goals for our virtual piano teacher application to achieve. We intended our piano teacher to address the various deficiencies of previous works by accomplishing the following:

- Not requiring the user to perform any calibration (e.g. ROI selection or entry of keyboard parameters).
- Allowing the keyboard to be partially obstructed by the player's hands at all times, at least as much as is expected during normal play.
- Not requiring a particular set of visible keys.
- Tolerating camera movement during use.
- Identification of both white and black keys.
- Not requiring a fiducial marker, other than the keyboard itself.
- Not requiring a camera that is more than a low-cost, low-quality webcam.

With these additional considerations in mind, what follows is the implementation of our virtual piano teacher divided into major steps.

A. *Equipment Setup*

Because our method requires no calibration, the setup consists only of connecting a webcam to the user's laptop and positioning it to view the piano keyboard; for example, by placing it atop the piano facing downwards, or attaching it to a stand next to the player. Once the virtual piano teacher is running, the next steps (for keyboard detection and key identification) are completed every frame.

B. *Conversion to Grayscale*

A piano keyboard is black and white only, so for keyboard detection and analysis color information is not needed. We create a copy of the current webcam frame and convert it to grayscale. We also blur the grayscale frame using a normalized box filter of five-pixel width and height. This is to reduce the

effects of noise. Fig. 1 demonstrates this with a sample video frame. We will use this same frame when illustrating each of the processing steps.

Figure 1. A frame of webcam video, converted to grayscale as the first step of processing. Note that the image is upside down as the camera has been placed on top of the piano facing down towards the player.

C. Keyboard Area Detection

To detect the keyboard area, a downsized 160×120 pixel copy of the grayscale frame is made to reduce processing time when searching for the keyboard. This small frame is then binarized, with an empirically chosen threshold of 150 (i.e. lighter pixels are set to 255 and darker pixels are set to 0). A Sobel, or more specifically a Scharr, operator is then used to detect horizontal edges.

Next, a Hough transform is used to detect the lines across the top and bottom of the black keys, as well as the line across the bottom of the white keys. A distance resolution of one pixel, an angle resolution of one degree, and an accumulator threshold of 50 was empirically found to detect suitable lines. As mentioned previously, we decided our method must work even when the player's hands are partially obscuring the keyboard. The accumulator threshold of 50 was found to result in the keyboard lines being detected, even when a hand blocked edges from being detected by the Scharr operator.

Because the Hough transform detects lines in all directions, we iterate through the detected lines and reject any that are not within ±5 degrees of horizontal. This limits the keyboard to being approximately parallel to the frame's upper and lower edges. Ideally, there would now remain only three lines – one each across the top, bottom and middle of the keyboard[1].

However, due to the fish-eye distortion of the webcam used, the Hough transform may sometimes find several lines of similar gradient where only one is desired. To rectify this, our method sorts the lines into three clusters based on the relative distances between their mean y-coordinates. The clustered lines are then averaged, giving the three lines defining the keyboard area of the frame.

[1] These three lines are, in our method, all which define the boundaries of a keyboard. The lines extend from the left of the frame to the right, and so the left and right ends of the detected keyboard are simply considered to be the outsides of the first and last detected keys.

Fig. 2 shows the lines, found through this process, overlaid on the output frame – a debug option of our application.

D. Frame Rotation

Because the camera can be positioned above the piano (looking down and towards the keyboard and user), or can be placed to view the keyboard from the user's perspective, the images of the keyboard taken by the webcam may or may not be upside-down.

Our program examines the small, binarized frame to determine the orientation of the detected keyboard. For each black pixel between the top and bottom lines, a count is incremented depending on whether the pixel is above or below the middle line. If there are more black pixels above the middle line, then the keyboard is the right way up – otherwise it is upside down and the frame is rotated by 180 degrees.

From this point forward, all processing is done with the frames oriented to show the keyboard the right way up. Fig. 2 shows our sample frame with this correction applied.

E. Key Detection

Our key detection algorithm involves traversing a line across the top of the keyboard. This gives a sequence of white and black pixels which we use to determine the location and name (e.g. A, B-flat, B etc.) of each key. We choose the top of the keyboard for this because a player's fingers do not enter this area during normal playing. Furthermore, when keys are pressed, the depression of the keys is significantly less visible at this end.

To begin key detection we select a region of interest in our full-sized, grayscale frame. The region begins at the top line of the keyboard, and extends down for a proportion of the keyboard height. We then threshold this region of the frame only, and traverse a line parallel to the top of the keyboard and slightly below it. Whenever a black-to-white or white-to-black pixel transition is discovered, its location in the frame is recorded as a potential key edge.

The lines created by the gaps between consecutive white keys are not reliably detected as potential key edges in certain cases. This is worsened in bright lighting conditions, or when the camera is positioned very highly above the keyboard. For

Figure 2. The result of the keyboard detection algorithm. Candidate lines are shown in green, with the final three (averaged) lines shown in red. The frame has been rotated 180° for ease of playing.

this reason we ignore any edges around black regions which are too small. What we are left with now are edges between black and white keys only, as well as (possibly) some edges beyond the ends of the keyboard. We now use the dimensions of standard piano keyboards given by Savard [14]. From this we know that both black and white keys are approximately the same width at the top of a keyboard.

We now find the distance between each pair of edges, which corresponds to the widths of the tops of potential keys. The most commonly occurring width (to the nearest 10 pixels) is found, and the mean of all occurrences of this width is assumed to be the true width of the top of a key. Because we now have the true width of the key-tops, in image coordinates, we iterate through each pair of edges once more – examining the distance between each pair. The first distance comparable to a key-top width marks the leftmost key, and each subsequent distance, of the correct size, marks another key. The color of the key is determined by the type of the left edge (i.e. a black-to-white edge marks a white key and vice versa).

This continues until the edges run out, or a potential key width is too large compared to the known key-top width. The one exception is when a white key's width is found to be approximately twice the expected size; this marks a pair of consecutive white keys – either B and C or E and F. Fig. 3 shows the results of this process when applied to the example frame.

All that remains in the key detection phase is to find the boundaries of each key. We take our thresholded region of interest and apply a Scharr operator once more, this time to find vertical edges. From the vertical edges found, a Hough transform is used to detect the lines on either side of each key. The lines are matched to the key edges used previously, based on the shortest distance from each key edge to each line. If a line is not found close enough to a key edge (perhaps at the edge of the frame or between consecutive white keys), a line is interpolated based on neighboring keys.

In our approach, a key is defined as four corner points (representing a bounding quadrilateral) and a name. For the black keys, we calculate the corner points as the intersection of the left and right lines of the key with the top and middle lines of the keyboard.

Figure 3. Here we have found the location of the top of each key. Yellow is used to show where our method has detected white keys, and purple lines show black keys. Orange lines mark gaps between consecutive white keys.

For the white keys the process is more complex, as the tops of the white keys are narrowed due to the presence of the black keys. Fortunately, the keyboard dimensions given by Savard [14] allow us to calculate the corner points instead. We have now detected each of the keys. The method we have used is illustrated in fig. 4 through the use of another debugging setting in our application.

Figure 4. The final result of our key detection. The bounding boxes for white keys are shown in yellow, while black keys are shown in purple.

F. Key Identification

The final step of our keyboard detection and identification is to determine the names of each key we have found. From our previous step we have an unbroken sequence of detected keys; however we do not know how to distinguish between keys other than by their color. We do not even know if the keys we have detected form a valid sequence. Both of these issues can be resolved in the same way:

A valid sequence of keys, on a real piano, alternates between black and white with a unique pattern. If a black key is represented as a zero, and a white key is represented as a one, a valid, infinitely long sequence of keys can be represented as:

$$K_V = \{1, 0, 1, 0, 1, 1, 0, 1, 0, 1, 0, 1, ...\} \quad (1)$$

Here the valid sequence of keys, K_V, starts on a C. To test a key sequence of unknown validity, K_U, we take the discrete cross-correlation of the two sequences, defined as:

$$(K_V \star K_U)[n] = \sum_{m=-\infty}^{\infty} K_V[m]\, K_U[n+m] \quad (2)$$

For our example frame, it can be seen that there are 17 elements, representing keys, in K_U. Of the 17 keys, 10 are white (and so are represented by a one). In practice, K_V need not be infinitely long. A length of K_V equal to 11 (the period of K_V minus one) more than K_U is sufficient to allow a complete set of cross-correlation values to be obtained.

To find if a sequence of detected keys is valid, the cross-correlation is calculated. The cross-correlation is maximized when the valid sequence and unknown sequence are closest to matching. Furthermore, if this cross-correlation value is equal to the number of white keys in the unknown sequence, then the unknown sequence is valid for a real piano keyboard. The value of n which results in the maximization of the cross-

correlation is then used to determine the name of the starting key of the now known sequence. It is then trivial to completely determine the names of all the keys in the sequence.

G. *Virtual Piano Teacher*

In the above steps we have demonstrated a method for identifying the keyboard and keys of a piano. All that remains to complete our virtual piano teacher is to provide a means of interacting with the user. We allow the user to interact with our application through a graphical user interface.

The user selects a highlight mode, either *Chords* or *Scales*, and selects a piano key to be the root or tonic note for the chord or scale. The identification steps from above are used to highlight the appropriate keys for the selected chord or scale, and this is delivered as an overlay of the webcam footage on the user's computer screen. In addition to the highlighting of keys, we also provide fingering information to the user by drawing a number above each key in the selected chord or scale.

The numbers one through to five are used for the fingering prompts, starting from the thumb and ending at the small finger; this notation is a commonly used convention. There are no formal rules for piano fingering, however there are informal rules that are most commonly adhered to – these exist for both chords [15] and scales [16, 17].

IV. Results and Discussion

To evaluate our proposal for a webcam-based virtual piano tutor, we must look to the original goals we aimed to achieve with the system. In addition to those listed at the beginning of Section III, these were to:

1. Propose a virtual piano teacher which can be used to teach fundamental chords, scales and suitable fingering to a beginning pianist.
2. Require only a typical laptop and a low-cost webcam.
3. Teach through an augmented reality style display, which requires the application to run in real-time.

In this section we reflect on these goals in order to determine the level to which we succeeded in our proposal. Fig. 5 shows the completed software application.

A. *Performance*

Our virtual piano teacher was developed and tested on a 2.20GHz dual-core laptop, with 4GB of RAM, and we used an NZ$13 no-brand webcam in order to develop for the lowest common denominator of devices. The webcam operates at a maximum frame rate of 15 frames per second.

The application itself makes use of the OpenCV computer vision library for its various image processing algorithms [18]. We have written the application in a mixture of native C++ and C++/CLI – Microsoft's C++ compatible language intended for use with the .NET Framework. This choice of language was made as it allowed the use of both OpenCV (written in C/C++) and Microsoft's Windows Forms GUI API.

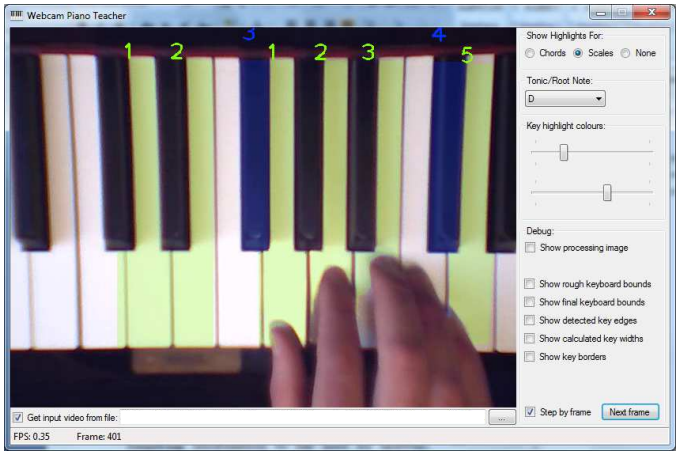

Figure 5. The virtual piano teacher application. On the left is the main display area for the video processed by the program. The right of the window contains the controls for selecting different chords and scales to learn, as well as other configuration options. At the bottom of the window the user can select the input source to be a webcam, or a video file – as is the case in this image where the user has chosen to learn the D major scale.

We tested the software's performance using the aforementioned webcam as a source of video. Rather than testing with live data, we used the webcam to record video to a file and then fed this as the input to our virtual piano teacher. The reason for doing this was to both provide consistent results for debugging, as well as allow the program's frame rate to be limited only by its performance and not by the webcam.

The result of the benchmarking with this file (the same file shown in all of the figures of this paper) was an average frame rate of 25.1 frames per second over the 702 frames of the video. We found that a significant amount of the processing during each frame was spent drawing the overlaid information to the output video – this accounted for approximately 19% of the processing time each frame.

Without drawing the highlights to the screen our application was able to reach an average frame rate of 31.2 frames per second – suggesting significant room for improvement in the drawing code of our program. Furthermore, prior to implementing the latter half of our key detection code, our application was able to reach average frame rates of between 55.0 and 65.0 frames per second. Considering that our key detection code involves the use of similar algorithms to our keyboard detection code, it is clearly possible to optimize our program for higher frame rates.

As it stands now, our virtual piano teacher is still well suited to real-time use. The current per-frame processing, including drawing code, results in an average delay of only 40 milliseconds between receiving a frame from the camera, and providing the augmented reality output to the user.

B. *Goals Met*

With respect to goal one, from the beginning of this section, we can say that we have demonstrated a learning tool that can teach all three of chords, scales and fingering to a beginner pianist. Currently our virtual piano teacher only supports major chords and scales; however it would be almost trivial to extend

our application for others (such as minor scales, diminished chords, etc). Regarding goal two, through our performance benchmarking we have demonstrated that our application does indeed require only a typical laptop and webcam. The webcam, in particular, has the lowest capabilities of any that a user would generally find. We can say that we have met goal three also, as our application is augmented reality in design and can perform suitably for real-time use. However, as is explained above, there is certainly room for improvement in our application's performance. In addition, the accuracy of our augmented reality display is far from ideal. As can be seen in fig. 5, there are keys whose boundaries have not been accurately determined, which results in the wrong keys being partially highlighted. Shadows also cause issues, in that they lead to the detected area of black keys being much larger than is correct in some cases. The user's hands can also be problematic, as certain lighting conditions can make the hands appear white. This causes the key highlights to be drawn over the user's hands.

The remaining requirements, from Section III, are all rather straightforward and have been met by our virtual piano teacher.

V. Future Work

The key areas to target for future work are those identified in the previous section as being deficient in some way.

An important area to improve upon is the performance and efficiency of the program. It should be possible to make changes in the key detection and identification code that will bring the average frame rate closer to 60 frames per second. The drawing code was another inefficient aspect of the software. Currently the drawing of highlights is completely performed on the CPU on a pixel-by-pixel basis; offloading this task to the GPU would be a significantly better solution. As the previous section mentions, the virtual piano teacher currently only supports major chords and scales. Extending the range of chords and scales supported would be beneficial to the application's use as an educational tool.

Perhaps the most important area of future work is on improving the overall robustness of the application and its key detection method. Finding more accurate key boundaries, and coping with shadows and lighting, would greatly improve the quality of the program. Reworking the software to use 3D registration as per the methods proposed by Barakonyi and Schmalstieg [12], and Huang et al. [3], may help in achieving this improved quality of operation.

VI. Conclusion

In this paper we have proposed a webcam-based virtual piano teacher application. This has included the proposal of a method for identifying a piano keyboard, and its individual keys, in video sourced from an inexpensive webcam.

Our implementation has successfully produced an application which can be used to teach a beginner pianist fundamental practical skills; that is, how to correctly play chords and scales with appropriate fingering. The application runs efficiently enough to be used in real-time, and achieves an average frame rate of 25.1 frames per second with typical hardware. Our program was developed with the limitations of previous work in mind – and these limitations were the basis of many of the goals of this paper. The goals have largely been met, demonstrating that this paper has been successful in addressing the limitations of previous work. We have devised a method which does not require any calibration or configuration on the part of the user; a convenient feature which was not present in any of the existing methods.

Despite the improvements we have made in some areas, our method has several limitations itself. Future areas of work have been identified as ways of improving our solution, with the most important of these areas being related to improving the accuracy of our method under different conditions.

References

[1] E. Dawe and R. Green, "Computer Vision Piano Tutor," Canterbury University 2010.

[2] C. Deaker and R. Green, "A Computer Vision Method of Piano Tutoring, Without the Piano," Canterbury University 2011.

[3] F. Huang, Y. Zhou, Y. Yu, Z. Wang, and S. Du, "Piano AR: A Markerless Augmented Reality Based Piano Teaching System," in *Intelligent Human-Machine Systems and Cybernetics (IHMSC), 2011 International Conference on*, 2011, pp. 47-52.

[4] D. O. Gorodnichy and A. Yogeswaran, "Detection and tracking of pianist hands and fingers," in *Computer and Robot Vision, 2006. The 3rd Canadian Conference on*, 2006, pp. 63-63.

[5] A. Oka and M. Hashimoto, "Marker-less piano fingering recognition using sequential depth images," in *Frontiers of Computer Vision, (FCV), 2013 19th Korea-Japan Joint Workshop on*, 2013, pp. 1-4.

[6] C.-H. Yeh, W.-Y. Tseng, J.-C. Bai, R.-N. Yeh, S.-C. Wang, and P.-Y. Sung, "Virtual Piano Design via Single-View Video Based on Multifinger Actions Recognition," in *Human-Centric Computing (HumanCom), 2010 3rd International Conference on*, 2010, pp. 1-5.

[7] K. Huang, E. Y. L. Do, and T. Starner, "PianoTouch: A wearable haptic piano instruction system for passive learning of piano skills," in *Wearable Computers, 2008. ISWC 2008. 12th IEEE International Symposium on*, 2008, pp. 41-44.

[8] M. Lang and R. Green, "Musical Desktop: A Webcam Piano," Canterbury University 2012.

[9] C.-C. Lin and D. S.-M. Liu, "An intelligent virtual piano tutor," presented at the Proceedings of the 2006 ACM international conference on Virtual reality continuum and its applications, Hong Kong, China, 2006.

[10] M. Sotirios and P. Georgios, "Computer vision method for pianist's fingers information retrieval," presented at the Proceedings of the 10th International Conference on Information Integration and Web-based Applications & Services, Linz, Austria, 2008.

[11] C. v. Hardenberg and F. Bérard, "Bare-hand human-computer interaction," presented at the Proceedings of the 2001 workshop on Perceptive user interfaces, Orlando, Florida, 2001.

[12] I. Barakonyi and D. Schmalstieg, "Augmented Reality Agents in the Development Pipeline of Computer Entertainment," in *Entertainment Computing - ICEC 2005*. vol. 3711, F. Kishino, Y. Kitamura, H. Kato, and N. Nagata, Eds., ed: Springer Berlin Heidelberg, 2005, pp. 345-356.

[13] P. Lamb. (N.D., 04/05/2013). *ARToolKit* [Online]. Available: http://www.hitl.washington.edu/artoolkit/

[14] J. Savard. (N.D., 04/05/2013). *The Size of the Piano Keyboard* [Online]. Available: http://www.quadibloc.com/other/cnv05.htm

[15] B. Kraemer. (N.D.). *Major Piano Triad Chords* [Online]. Available: http://piano.about.com/od/chordskeys/ss/major_triads_treble.htm

[16] R. Kelley. (2001, 04/05/2013). *Scale Fingering Chart* [Online]. Available: http://www.robertkelleyphd.com/scalfing.htm

[17] N. Wickham. (2010, 04/05/2013). *Free Piano Scale Fingering Diagrams* [Online]. Available: http://musicmattersblog.com/2010/04/06/free-piano-scale-fingering-diagrams/

[18] OpenCV Developers Team. (N.D., 04/05/2013). *About OpenCV* [Online]. Available: http://opencv.org/about.html