An Image Analysis Approach for Transcription of Music Played on Keyboard-like Instruments

Souvik Sinha Deb
Indian Institute of Technology Bombay
souviksinhadeb@gmail.com

Ajit Rajwade
Indian Institute of Technology Bombay
ajitvr@cse.iitb.ac.in

ABSTRACT

Music transcription refers to the process of analyzing a piece of music to generate a sequence of constituent notes and their duration. Transcription of music from audio signals is fraught with problems due to auditory interference such as ambient noise, multiple instruments playing simultaneously, accompanying vocals or polyphonic sounds. For several instruments, there exists added information for music transcription which can be derived from a video sequence of the instrument as it is being played. This paper proposes a method to utilize this *visual* information for the case of keyboard-like instruments to generate a transcript automatically, by analyzing the video frames. We present encouraging results under varying lighting conditions on different song sequences played out on a keyboard.

CCS Concepts

•**Computing methodologies** → *Activity recognition and understanding;*

Keywords

Music transcription; Computer Vision Applications

1. INTRODUCTION

Music transcription is defined as the process of noting down music that has already been played or recorded, in the form of a sequence of notes (i.e. the fundamental frequency of the sound signal) and their time duration (difference between the end and start time of the note). The major applications of music transcription include digital rerendering of music using the sounds of other instruments, editing of the musical piece, or analysis (*e.g.*, determining the degree of similarity between two musical pieces, or searching a database of musical transcripts for the presence of a certain musical phrase). It can also be of pedagogical value and help students and teachers of music in the study of certain musical forms.

ICVGIP, December 18-22, 2016, Guwahati, India

DOI: http://dx.doi.org/10.1145/3009977.3010007

There are two different types of musical systems - monophonic and polyphonic. Monophonic digital music is produced when there is only one instrument playing one note at a time. While pitch detection for monophonic systems is a widely solved problem, in the case of polyphonic digital music, which may contain multiple instruments along with vocals all playing at the same time, it still remains largely unsolved [3, 4, 8]. It is known that the varied forms of western music are primarily polyphonic, whereas the Indian classical traditions are monophonic. However even in such monophonic traditions, it is not uncommon to have multiple melodies being played simultaneously. Examples include the background drone or *tanpura* in Indian classical concerts (both vocal and instrumental), the playing of multiple keys simultaneously on a harmonium to generate interesting melodic effects, or simply multiple musical instruments playing together. These scenarios present further challenges for music transcription even in primarily monophonic traditions.

Due to these difficulties, one may resort to a hardware-based solution independent of audio analysis. Some instruments have a MIDI (Musical Instrument Digital Interface) which allows one to electronically store the output of the instrument in a computer. MIDI is common in electronic instruments such as the synthesizer (also called 'keyboard') or the electronic piano. However MIDI is not present in several types of pianos or organs (and some keyboards as well), and in all Indian instruments such as the harmonium or the *sitar*. For such cases, automatic music transcription from live concerts or an informal 'jam session' becomes challenging due to the difficulties that arise in performing source separation from a single recording. The problem is further exacerbated by ambient audio noise such as audience applause or people speaking. While noise cancelling microphones can be employed for eliminating background noise [6], they are not effective in eliminating unwanted sounds of high amplitude, or the effect of other instruments or singing voices nearby. One could imagine the use of dedicated 'shotgun' microphones[1] for each instrument, each having a very narrow angle of acceptance (also called pickup angle). This imposes new challenges such as synchronization of all such microphones, higher costs, and new difficulties in dealing with large-sized instruments such as the piano. However, the biggest difficulty in their usage lies in the separation of multiple sounds from the same instrument: for example, the sound of two or more keys being pressed simultaneously on

[1] http://www.audio-technica.com/cms/site/62073812c42084c8/

Figure 1: Keyboard-like instruments without MIDI: harmonium (top row, left), organ (top row, right), grand piano (bottom row). Image sources: wikipedia articles on harmonium, reed organ and grand piano.

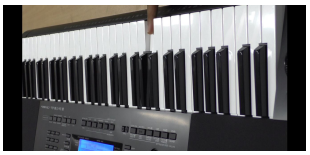
a harmonium or a piano. In such scenarios, information in the form of a video where all the sources of sound are clearly visible can be of immense help. In this paper, we present the relatively unexplored idea of using image processing to detect the actual keystrokes for a piano, harmonium or keyboard (called 'keyboard-like instruments' - see Figure 1), in order to automatically generate a transcript. We present results for song sequences played out on a keyboard, as it allows for comparison with the ground truth generated in the MIDI file. However, our method can be easily extended to the harmonium or the piano, and it may ultimately be scaled to other instruments such as a *sitar*, and can be deployed individually for all instruments in a concert thereby auto generating the whole transcript. Such an image-based technique is inherently immune to ambient audio noise or the interfering effects of multiple instruments.

This paper is organized as follows. In Section 2, we describe our experimental setup to record video sequences of a keyboard while it is being played. Section 3 describes the preprocessing steps and the main image analysis method. Experimental results are presented in Section 4, the relation between our approach and some existing literature is discussed in Section 5, and a discussion/conclusion follows in Section 6.

2. EXPERIMENTAL SETUP

Our experimental setup consisted of a keyboard besides a video camera on a tripod. Most of our initial experiments were performed on a Casio CTK860IN keyboard as it is MIDI enabled which allowed us to generate ground truth for comparison. A high definition Panasonic HC-V750 camera capable of video acquisition at a 1920 x 1080 resolution with 120 fps was used. But for this particular problem we obtained satisfactory results with videos captured at 640 x 480 resolution with 28 fps without encountering any motion blur artifacts. The camera was positioned on the left side of the keyboard at a height of 3 feet above it. An orthographic view was deliberately avoided because in such views, the changes in the images due to a key-press were almost negligible. The side view was better suited to capture changes in key structure during key-presses and hence the apparatus was arranged in this manner as shown in Figure 2.

(a) Apparatus



(b) Frames acquired by camera

(c) Geometrically normalized and cropped frame

Figure 2: Setup of apparatus for music transcription

3. PREPROCESSING AND IMAGE ANALYSIS

The aim of our image analysis was to determine which key(s) was (were) pressed in *each* frame of the video. This created a sequence of notes played and determined their duration as well, since the frame rate of the video camera was known. Hence, this was used to generate a MIDI file which can be played on a computer or a keyboard. However before such analysis can be performed, there were some preprocessing steps required.

3.1 Geometric Normalization

The camera captures a wide range of keys and other irrelevant portions of the keyboard or background in each frame (see Figure 2). It was decided to restrict key-press detection to only two octaves of the keyboard, ranging from the lower C (key #48 as per MIDI format) through the middle C to the upper C. Such a range is generally the norm in Indian classical music. A geometric normalization was performed to align the acquired frames with an orthographic view of the keyboard as the latter is more convenient for preprocessing. For this, four points on the plane containing the white keys of the keyboard were manually selected in the first frame of the video to serve as boundary points of the relevant octaves.

They were mapped to corresponding corners of a reference image captured in an orthographic view and a homography mapping was calculated. This selection of points had to be performed once on *only the first frame* as all subsequent frames were subjected to the same mapping. Prior to the homography computation and transformation, the images were converted to gray-scale, and cropped beyond the selected boundaries. One such frame after the transformation is shown in Figure 2(c).

There was an implicit assumption that the keyboard as well as the camera would be stationary throughout the recording, which was true for all our experiments. However, the frame alignment problem in cases where the keyboard moves during the recording can be easily dealt with by placing a few flourescent markers at various corners of the keyboard (although we did not use this in our experiments). This would in fact avoid the manual step altogether as well.

3.2 Key Boundary Extraction

Next, the step boundaries of each of the white and black keys were automatically calculated. For each black key, all pixel intensity values along a horizontal scan-line over the black key were stored separately. Values greater than a prefixed mid-range value (between black and white) were assigned the highest value and those lower were assigned the lowest value from the allowable intensity range. Now all transitions in the line from high to low or vice-versa signaled a boundary for the black key. The specific location of the scanline as well as the specific mid-range value was observed to be not very important in our experiments.

For the white keys, all edges were detected in the white key area by using a Canny edge detector. Hough lines were calculated from the edges and all approximately vertical and sufficiently long lines were treated as white key boundaries. In some cases, this procedure was seen to miss a few key boundaries (especially under bright lighting). But the missing boundaries were easily extrapolated as all white keys have equal width and the homography maps the side view to an orthographic projection. Assuming that more than half of the boundary lines were properly detected, the median of the widths of all the detected keys was set to be the actual key-width and erroneous boundary lines were corrected. The computed boundaries for both black and white keys are shown in Figure 3(a).

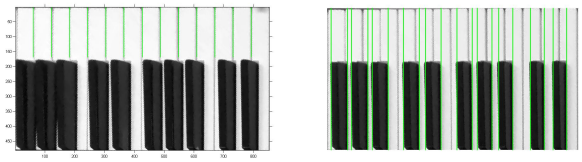
3.3 Key-press Detection

Since the geometry of the white and black keys is very different, both these cases were considered separately.

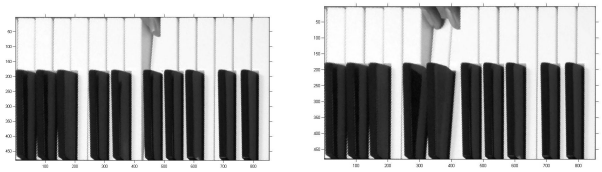
White keys: Figure 3(b) shows the transformed image of the keyboard with pressed white keys. It was observed that a single pressed white key always resulted in baring the usually covered side of its adjacent white key on the right side (as the camera is placed on the left side of the keyboard). This was an important characteristic to identify pressed white keys. We attempted to design a simple rule-based algorithm to detect this pattern and to mark the corresponding white key as pressed by creating a set of handcrafted rules. However, problems arose due to variations in the structure of this pattern due to changes in lighting. Also this particular pattern was markedly different for two adjacent key-presses as compared to a single key-press as seen in Figure 3(b).

In order to design a general and robust method for this problem, we decided to resort to a machine learning method. A support vector machine (SVM)[2] was trained to classify each key as pressed or not pressed instead of creating a manual combination of heuristic rules. The training was performed on a set consisting of feature vectors extracted from non-pressed keys (labeled '-1) and single pressed keys (labeled as '+1') as well as two adjacent pressed keys (both labeled as '+1'). The feature vector consisted of a part of the scan-line covering the width of the relevant key and half of its two adjacent keys (see Figure 3(d)). The scan-line was taken roughly midway between the edge of the white keys and the edge of the black keys, but its exact position did not have any impact on the results. Note that this classifier was trained to detect single key-presses as well as two adjacent white key-presses.

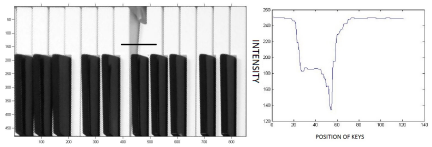
Black keys: Detection of black keys posed a different challenge as the distinctive pattern in pressed white keys is absent in pressed black keys. Also some visible parts of the black keys do not lie in the same plane (since the camera is placed on one side of the keyboard as in Figure 2) and so a single homography fails to produce a proper orthographic view of the black keys. Figure 3(d) depicts that the size of the keys on the left side (closer to the camera) of the image is



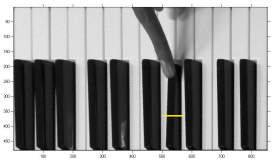
(a) Boundaries for black and white keys (marked in green)



(b) Pressed white keys: single and two adjacent



(c) Feature vectors for white keys



(d) Feature vectors for black keys

Figure 3: Various steps for preprocessing before key-press detection

[2]https://www.csie.ntu.edu.tw/~cjlin/libsvm/

much wider than that on the right side. However, black keys when pressed go deep into their own grooves reducing the amount of black pixels in the vicinity of their boundaries. This was exploited to design feature vectors for the keys. Similar to the white keys, scan-lines taken from the vicinity of the black keys were chosen to be the feature vectors of that particular key, as shown in Figure 3(d), except that here the scan-lines were resized to a fixed size (60×1) using interpolation. Whenever a key was pressed the amount of black pixels in the scan-line receded and produced a new pattern. However since different black keys had different apparent width in the acquired images, the patterns of these vectors were different for each individual key. Hence we divided the black keys across the two octaves into two different groups based on their position (the first 7 keys nearer to the camera for the first group and the last 3 keys in the second group, out of the total of 10 black keys in the chosen two octave range), and trained a *different* SVM for each group. Since for black keys, a single homography is an approximation, the apparent width of the black keys in the geometrically normalized image undergoes minor variations w.r.t. the camera viewpoint. However, we observed that our SVM-based classifiers were robust to those variations.

3.4 Illumination Normalization

The feature vectors described above are not invariant to changes in illumination. To bring about this invariance, we adopted the method in [10] where it was applied to face images. The method is essentially a type of homomorphic filtering pivoted around the assumption that the intensity profile I of a surface is determined by its inherent albedo ρ and external lighting. Under the Lambertian assumption, we have $I = \rho \boldsymbol{n} \cdot \boldsymbol{s}$ where $\boldsymbol{n}$ and $\boldsymbol{s}$ stand for the surface normal direction and the lighting direction respectively. Since the keyboard surface is primarily flat, we have $\nabla I \approx \nabla \rho W$ where $W = \boldsymbol{n} \cdot \boldsymbol{s}$ is a constant. To remove the effects of illumination which is dominated by lower frequencies, we use a smoothed version of I given as $I_s = I * g$ where g is a Gaussian kernel. This yields us a normalized gradient given by $\nabla \hat{I} = \dfrac{\nabla I}{W} \approx \nabla \rho$. Now, $\nabla \hat{I}$ is basically a high-pass filtered version of I and it may be very noisy. To remove the noise and restore the structural pattern, an integration operation is performed on these gradients to yield an illumination invariant representation. For this, we employed the method from [1] based on the popular method by Frankot and Chellappa. The effect of this illumination normalization step is clearly depicted in Figure 3.4. It should be noted that this technique is related to the retinex algorithm [7]. This technique is largely unaffected by cast shadows except along the boundaries of the cast shadow.

3.5 Transcript Generation

The SVMs for black and white keys were trained offline on scan-lines extracted from geometrically- and illumination-normalized frames acquired under six different lighting conditions. The scan-line for each key in each frame was manually labeled as 'key-press' or 'key not pressed', which produced the ground truth for training. Now for testing, each frame of a new video to be transcribed was similarly normalized as described earlier, and the scan-line at each key in each frame was classified using the SVM. This produced the sequence of notes. The duration of each note was computed from the frame indices and the (known) frame-rate of the camera. A MIDI file was generated from this information using publicly available software[3].

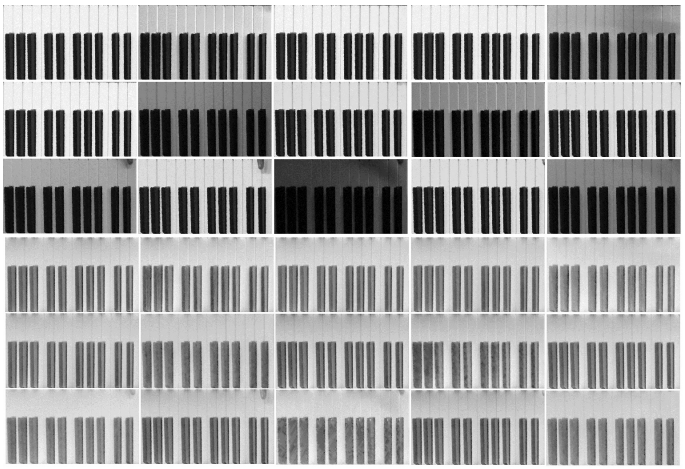


Figure 4: Original frames (top 3 rows) and their corresponding illumination-normalized frames (bottom 3 frames)

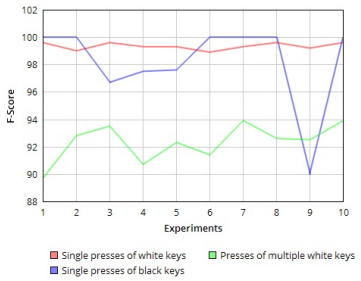
4. EXPERIMENTAL RESULTS

All experiments were carried out indoors in a lab with several tube-lights fixed to the ceiling.

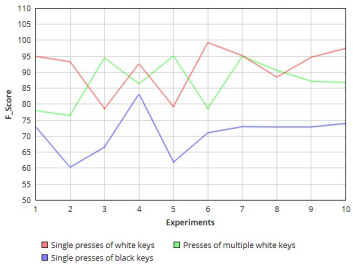
Our first set of experiments were carried out to test the efficacy of the key-press detection algorithm. For this, we recorded videos of a keyboard while an amateur player was striking only single white keys of the keyboard, or single black keys, or a pair of adjacent white keys, on a keyboard. Every frame was manually tagged as either 'no key-press' or else with the number(s) of the keys that was (were) pressed. The SVM classifiers were tested on other similar videos of the keyboard, under slightly different lighting conditions simulated by slightly changing the position of the keyboard. The performance was tested separately for single white keys, double adjacent white keys and black keys using three performance measures: accuracy (fraction of correct classifications for each frame), precision (number of correctly detected key-presses/total number of detected key-presses) and recall (total number of detected key-presses/total number of actual key-presses). Finally, the F-score, i.e. the harmonic mean of the precision and recall, was calculated. As can be seen in Figure 5(a), for single black and single white key-presses the F-scores were all between 99-100%, whereas for two adjacent white key-presses, these parameters were around 93%.

A second set of experiments were performed on a dataset gathered under 15 different lighting conditions within our laboratory (see Figure 3.4 for a sample of lighting conditions). The SVM classifiers were learned on feature vectors extracted from geometry- and illumination-normalized frames taken from a subset of six illumination conditions and tested on the remaining nine. This was repeated ten times, using a different subset of six for training each time. The F-scores results are plotted in Figure 5(b). It should be noted

[3]http://kenschutte.com/midi



(a) Minor illumination variation



(b) Major illumination variation

Figure 5: Performance measures in the form of F-scores for key-presses under minor and major illumination variation

that these results are on a frame per frame basis. The recall is much higher if the percentages are computed by counting the key-presses instead of the number of frames. This is because we almost *never* missed an entire key, though we missed some frames from the key-press.

A third set of experiments was performed on a set of eight song sequences played on the keyboard by an amateur player. The duration of each song was around 20 seconds. Six of the songs consisted of snippets played out in the following Indian classical music scales (*raagas*): *Durga*, *Malhar*, *Bibhas*, *Bhupali* and *Megh*, two were pieces from two popular songs: 'Twinkle twinkle little star', and *tomaar holo shuru* (a popular Bengali song), and the last one consisted of all consecutive seven sharp notes played out in sequence (this is called as *sargam* in Indian music). The classification was performed using the SVM classifiers learned in the previous (*i.e.*, second) set of experiments, and transcripts were generated in the form of MIDI files. The generated MIDI files were played using a software called Anvil Studio 2015[4]. It was observed that the MIDI files generated by our program sounded similar to the original MIDI file recorded while the song sequence was being played, some errors due to false positives notwithstanding. In our transcription, we have artificially removed any period of silence at the beginning of the original song, and hence the transcribed files are usually slightly shorter than the original ones.

For reference, the original and reconstructed MIDI files and their corresponding conversions to mp3 format are included in the supplemental material accompanying this paper. The MIDI files as well as the mp3 files can be directly played out in Windows Media Player.

[4]http://www.anvilstudio.com/

Song	Percentage Accuracy
Malhar	91.05
Durga	95.95
Twinkle	70.87
Tomaar	89.37
Sargam	94.57
Megh	91.5
Bhupali	93.53
Bibhas	70.35

Table 1: Percentage accuracy of transcription for each song (measured w.r.t. manually marked out ground truth in each video frame)

Due to errors in transcription, it was very difficult to directly compare the MIDI files as a measure of ground truth. It was also difficult to compare the original and transcribed waveforms (*i.e.*, audio signals) due to their slightly different lengths. Hence, we measured the transcription quality in terms of the number of correctly classified frames in comparison to a manually marked ground truth in every video frame. A correctly classified frame was defined to be one of the following: (1) a frame in which no key was pressed and for which the classifiers all returned with a no-press, (2) a frame in which only one key was pressed, and for which the lone key-press was correctly detected with no other key-press detected, (3) a frame in which only one key was pressed and for which the key-press with the highest probability (as output by the SVM) was the correct one. Note, however, that in the actual transcription, we did *not* artificially eliminate such false positives that had a low probability as output by the SVM. Also note that in our experiments, we have not worked with multiple key-presses per frame. Based on this assumption and definition, the percentage of correctly classified frames for the eight songs are presented in Table 4. Resources pertaining to the experiments carried out can be accessed from https://www.cse.iitb.ac.in/~ajitvr/music_transcription.

Processing times: There are several steps of our algorithm, of which the illumination normalization and classification using the SVM are the most expensive. The processing time per frame was around 1 second in all on a standard PC, without exploiting any parallelization.

5. RELATED WORK

In this section, we summarize the very few existing research papers on this or related topics. The work in [11] uses video sequences with clear audio to learn cross-modal associations between audio and visual information. These associations learned offline are applied to video sequences of people speaking or the playing of a large xylophone using a hammer, to denoise the audio components corrupted by ambient noise using the (clean) visual information. This approach is called 'cross-modal denoising'. The image analysis problem for the xylophone in [11] does not consider illumination variations and it is a simpler task since the hammer hitting the xylophone is a feature that is more distinctive than a single or double key-press for a keyboard. The approach in [5] analyzes video sequences of a piano being played to detect and track the pianist's hands or fingers for the purpose of teaching fingering techniques to students remotely.

A technique for fingering recognition from depth images is proposed in [9]. None of these three techniques can be directly compared to our proposed method, as they work with essentially different problems.

Recently, a computer vision technique for music transcription from pianos was proposed in [2]. This technique detects key-presses using the difference images, i.e. the difference between the frames of the video sequence and a reference image that contains no key-presses. The main observation is that a white key-press is associated with a decrease in the intensity of the neighboring pixels at the base of the piano, as portions of nearby black keys which were not visible earlier are now exposed. This gives rise to negative-valued blobs in the difference image at the base of the piano. Similarly, a black key-press is associated with a positive valued blob in the difference image at the base of the piano. While this method has shown very beautiful results, the rule for reporting a key-press is based on the size of this blob. As such, this rule is not designed to handle illumination changes while the piano is being played. This is because such illumination changes will change the threshold required to decide the presence of a positive or negative blob since the reference image and the current video frames will have very different intensity values. In fact, these thresholds will not necessarily be constant throughout the image if the lighting change is non-uniform. This will require additional machinery for updating the reference image during the music recording or jam session. On the other hand, our approach attempts to normalize against major illumination changes, and the learned classifier is trained on images acquired under many lighting conditions.

6. CONCLUSION

We have proposed a very simple and easy to implement approach to transcribing music played out on keyboard-like instruments. The approach makes novel use of visual information. This is, to the best of our knowledge, the first technique that is trained to handle a variety of illumination changes, including those that occur during the musical performance (due to actual change of ambient lighting, as well as due to shadows cast if people move around during informal jam sessions), for the purpose of image-based music transcription. While we have encouraging results, our approach does get hindered by false positives due to issues as occasional occlusion of the keys by the hands of the keyboard player. This can be tackled by using two cameras, one on the left and one on the right side of the keyboard respectively, which we hope to accomplish in future. Future work will also involve more extensive transcriptions for music played out on pianos or harmoniums, as well as combining visual and auditory data for multi-modal transcription.

7. ACKNOWLEDGMENTS

The authors would like to thank Prof. Rushikesh Joshi, Prof. Bhaskaran Muralidharan, Prof. Preeti Rao and Prof. Sharat Chandran for very fruitful discussions and valuable suggestions.

8. REFERENCES

[1] A. Agrawal, R. Raskar, and R. Chellappa. What is the range of surface reconstructions from a gradient field? In *European Conference on Computer Vision*, pages 578–591, 2006.

[2] M. Akbari and H. Cheng. Real-time piano music transcription based on computer vision. *IEEE Transactions on Multimedia*, 17(12):2113–2121, Dec 2015.

[3] E. Benetos, S. Dixon, D. Giannoulis, H. Kirchhoff, and A. Klapuri. Automatic music transcription: Breaking the glass ceiling. In *ISMIR*, pages 379–384, 2012.

[4] E. Benetos, S. Dixon, D. Giannoulis, H. Kirchhoff, and A. Klapuri. Automatic music transcription: Challenges and future directions. *Journal of Intelligent Information Systems*, 41(3):407–434, Dec 2013.

[5] D. Gorodnichy and A. Yogeswaran. Detection and tracking of pianist hands and fingers. In *Canadian Conference on Computer and Robot Vision*, pages 60–69, 2006.

[6] R. Hendriks, T. Gerkmann, and J. Jensen. *DFT-Domain Based Single-Microphone Noise Reduction for Speech Enhancement - A Survey of the State of the Art*, volume 9 (1) of *Synthesis Lectures on Speech and Audio Processing*. Morgan & Claypool Publishers, 2013.

[7] R. Kimmel, M. Elad, D. Shaked, R. Keshet, and I. Sobel. A variational framework for retinex. *International Journal of Computer Vision*, 52(1):7–23, 2003.

[8] A. Klapuri. Automatic music transcription as we know it today. *Journal of New Music Research*, 33(3):269–282, 2004.

[9] A. Oka and M. Hashimoto. Markerless piano fingering recognition using sequential depth images. In *Korea-Japan Joint Workshop on Frontiers of Computer Vision*, pages 1–4, Jan 2013.

[10] S. Samsung. Integral normalized gradient image a novel illumination insensitive representation. In *IEEE Conference on Computer Vision and Pattern Recognition - Workshops*, pages 166–166, June 2005.

[11] D. Segev, Y. Y. Schechner, and M. Elad. Example-based cross-modal denoising. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 486–493, June 2012.