Guitar-Leading Band: Implementation Details

M. Cicconet, Visgraf/IMPA
L. Velho, Visgraf/IMPA
P. Carvalho, Visgraf/IMPA
G. Cabral, D'Accord Music Software

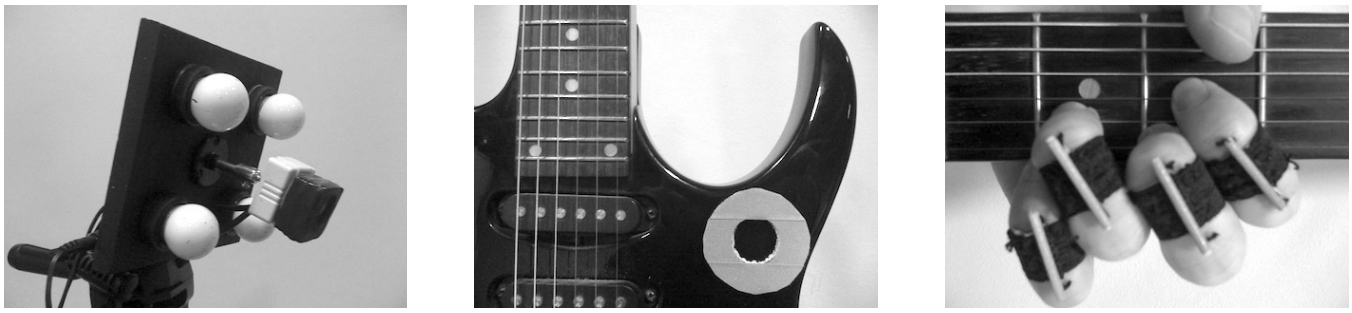

Figure 1: *Capture hardware. On the left, an infrared camera surrounded by four infrared light sources. In the center, a hollow disk made with reflexive material. Four of them are used to locate the plane containing the ROI. On the right, middle-phalanges gloves with small rods coated so as to easily reflect light.*

1 Video-Based Chord Recognition

We describe in this section the details of the guitar chord detection method based on the "visual shapes" of the chords, which are learned by a supervised Machine Learning method.

Let us define the Region of Interest (ROI) in the scene of a person playing guitar as being the region including the strings, from the nut to the bridge.

To facilitate the capture process, avoiding the overhead of segmenting the ROI, we chose to work in the infrared-light range.

Figure 1 shows the equipment that supports our method. We use a infrared camera to capture the scene, which is properly illuminated with infrared light. Special markers (fiducials) are attached to the guitar in order to easily locate the instrument, and for the fingers, reflexive gloves dress the middle phalanges.

The pipeline of our chord detection method is illustrated in Figure 2. The developed software takes advantage of some nice and robust algorithms implemented in OpenCV, an open source Computer Vision library [Bradski and Kaehler 2008].

First, a threshold is applied to the input image, so that the only non-null pixels are those of the guitar and finger markers. Then, using the contour detection algorithm and contour data structure provided by OpenCV, guitar and finger markers can be separated. Note that guitar fiducials and finger markers are, respectively, contours with and without a hole. Once the positions of the four guitar fiducials are known in the image, by using their actual positions in guitar fingerboard coordinates a projective transformation (homography) can be determined and applied in order to "immobilize" the guitar and easily extract the ROI. This homography is then applied to the north-most extreme of the finger rods, so we get the rough position of fingertips in guitar fretboard coordinates, since the distal phalanges are, in general, nearly perpendicular to the fingerboard.

We use a supervised Machine Learning technique to train the machine with the guitar chords we want it to identify. The chord a musician plays is viewed by the system as an eight-dimensional vector composed by the coordinates (after projective transformation) of the four fingertips, from the little to the index finger. We call this eight-dimensional vector the Visual Pitch Class Profile (VPCP), in analogy with an audio descriptor commonly used for chord recognition (the Pitch Class Profile [Fujishima 1999]).

Summarizing, the proposed algorithm for real-time guitar chord detection has two phases. In the first (the training phase), the musician chooses the chords that must be identified and takes some samples from each one of them, where by sample we mean the eight-dimensional vector formed with the positions of the north-most extreme of the finger rods, i.e., the VPCP. In the second (the identification phase), the system receives the vector corresponding to the chord to be identified and classifies it using the K Nearest Neighbor algorithm.

2 Automatic Composition

In this section we present the algorithm to generate music samples and the related probabilistic tools. In nutshell, a random sample of music is built using a Markov Chain conditioned to certain events. We have chosen to organize a music piece in cycles, bars and beats; and picked the major diatonic scale for the melody, since it is largely used in pop songs. Figure 3-top shows part of the diatonic scale notes as arranged on the guitar fretboard. Such an arrangement of notes is not algorithmically friendly. So we decided to use the representation shown in Figure 3-bottom.

The finite Markov Chain state-space is defined as $E = R \times M$. First, R is the space of rhythmic patterns. We have used five different states, corresponding to silence (rest), one whole, two halfs, three thirds and four quarter notes. Second, M is the space of possible notes, namely, the scale, whose states are the points of the matrix shown in Figure 3-bottom.

Each time a new beat is about to begin, a rhythmic pattern is sampled, depending on the current value of the parameter describing the "level of intensity" for the music. The greater the intensity, the greater the probability of playing more musical notes in the next beat. Once the rhythmic pattern is chosen, the melodic line is built. It is controlled by two independent Markov processes, one for the rows and the other for the columns of the musical notes, according to the representation of the diatonic scale shown in Figure 3.

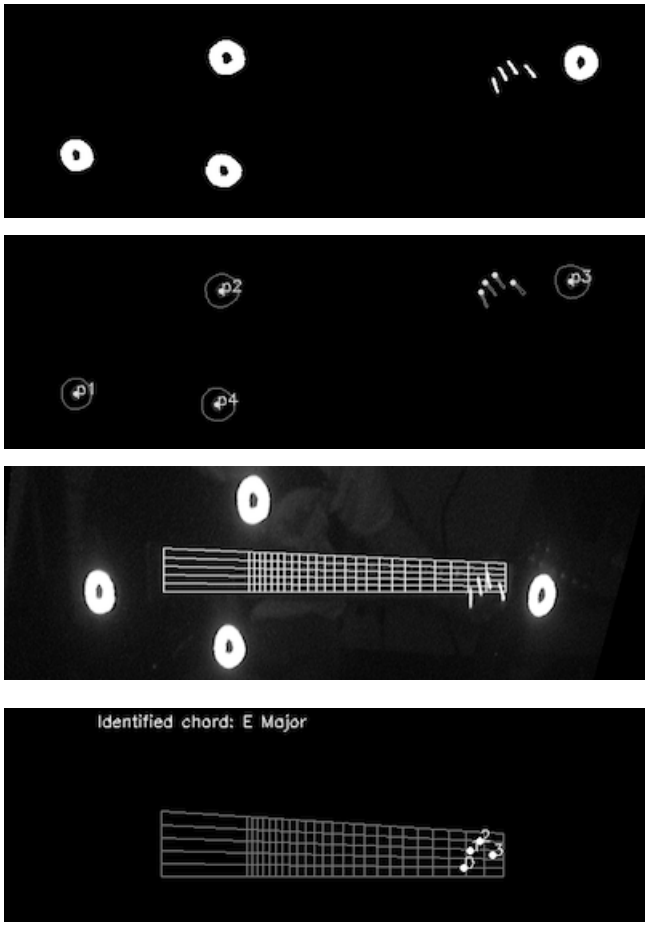

Figure 2: *Chord detection pipeline, from top to bottom. (1) A threshold is applied to take only guitar and finger markers. (2) Guitar fiducials and finger rods are detected using a contour detection algorithm. (3) A projective transformation "immobilize" the guitar, regardless the movement caused by the musician. (4) The projective transform is applied to the north-most extreme of finger roads in order to roughly locate the fingertips in guitar-fretboard coordinates.*

The conditioning on specific events mimics the behavior of a musician, who, when improvising, pursues a target note in meaningful chords. That is what can be called the *target-note* improvisation paradigm. As an example, the system may check if the first note of the sequence chosen for the beat is the same, regardless the octave, of the current chord's root note. Furthermore, the system may impose that the last note of the sequence should fall in some region of the matrix of musical notes. That region, for its turn, could be controlled by the location of the guitarist left hand in the guitar fretboard.

Let now A be the set of sequences which satisfy the just described restrictions. The method to simulate the conditioning of the Markov Chain on A is known as the Rejection Method, which consists simply in sampling a Markov Chain, and if the sample belongs to the set A, keeping it. If not, we resample until we get an allowed sample. Theoretically, the number of trials until an allowed sample to be obtained can be arbitrarily large. For this reason, we limited the number of trials. If no allowed sample is found, the last one is chosen. Of course doing this we do not simulate exactly the conditioned Markov Chain defined above. Nevertheless, this way the algorithm imitates musician's errors, when the target note is not reached, something that can eventually happen.

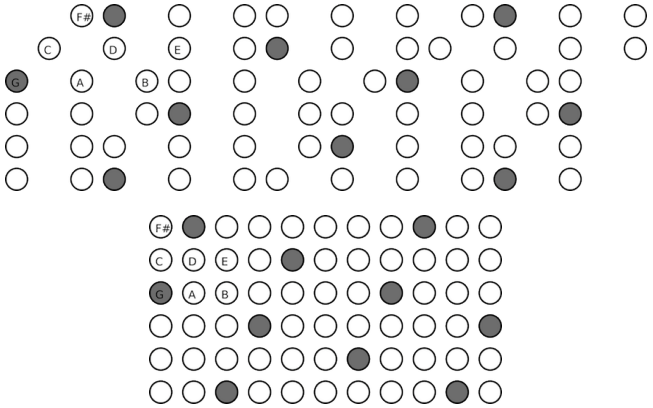

Figure 3: *Arrangement of notes from the diatonic scale in the guitar fretboard (top) and its representation as we have used to build the automatic composition algorithm (bottom).*

We have used the uniform distribution as initial distribution of the sequences for rhythmic patterns, (X_i), and melody, (Y_j). The transition probabilities for (X_i) depend on de current level of intensity of the song, as mentioned before. Regarding the melodic line, Y_j, the transition probabilities matrix may be set in a way such that the same note is not played consecutively, what gives more variability to the melody.

3 Plugging the Methods

Here we describe the pseudo-score of a music piece we have composed, at which the previous methods were used cooperatively.

The piece is organized in cycles, bars and beats. The progress of the piece in the course of the cycles is as follows:

- Cycle 1
  The strings ensemble follows the chords played by the musician, as recognized by the computer vision system.
- Cycle 2
  The drum loop number 1 is triggered by hitting the "Enter" button.
- Cycle 3
  Musician starts fingering, keeping the shape of the chords, so the video-based chord recognition algorithm can work properly.
- Cycle 4
  Musician starts strumming. Drum loop changes to level 2, a more intense level. Automatic composition algorithm starts at level of intensity 1.
- Cycle 5
  Automatic composition algorithm goes to level of intensity 2.
- Cycle 6
  Drum loop changes to number 3, the more intense level. Automatic composition algorithm goes to level of intensity 3, the greatest.
- Cycle 7
  The number of restrictions to be satisfied by the sequence of notes increases. The musician should play the sequence of chords that will be repeated for the next two cycles.

- Cycle 8
  Drum loop goes back to level 1. “Air Guitar” mode is turned on: the position of the hand indicates the region to which the improvised sequence of notes has to converge. Automatic composition algorithm goes back to level 2.
- Cycle 9
  The parameters of the previous cycle are kept. The system remains in the “Air Guitar” mode to give it significant importance.
- Cycle 10
  Control of the sequence of chords goes back to the computer vision system. Musician changes the guitar effect. “Air Guitar” mode is turned off. Automatic composition algorithm returns to level 3. Drum loop returns to level 3. Musician performs a sequence of chords different from the one of the previous cycle.
- Cycle 11
  Parameters of the system are the same as in cycle 10. Musician performs yet another sequence of chords.
- Cycle 12
  Parameters of the system are the same as in cycle 11. Musician performs the first part of the riff of chords preparing the conclusion of the piece.
- Cycle 13
  Parameters of the system are the same as in cycle 12. Musician performs the second part of the riff of chords preparing the conclusion of the piece.
- Cycle 14
  Drum loop goes back to level 1. Musician positions the hand in the last chord of the piece and performs the last strum. This is the last cycle of the music piece.

References

BRADSKI, G., AND KAEHLER, A. 2008. *Learning OpenCV: Computer Vision with the OpenCV Library*. O’Reilly.

FUJISHIMA, T. 1999. Real-time chord recognition of musical sound: A system using common lisp music. In *International Computer Music Conference*.