The Visual Keyboard: Real-Time Feet Tracking for the Control of Musical Meta-Instruments

Frédéric Jean [a,*] Alexandra Branzan Albu [b]

[a] *Dept. of Electrical and Computer Engineering Laval University, Québec, QC, Canada, G1K 7P4*

[b] *Dept. of Electrical and Computer Engineering University of Victoria, Victoria, BC, Canada, V8W 3P6*

Abstract

This paper proposes a new perceptual interface for the control of computer-based music production. We address the constraints imposed by the use of musical meta-instruments during live performance or rehearsal by tracking feet motion relatively to a visual keyboard. The visual attribute stands for the fact that, unlike its physical counterpart, our keyboard does not involve any force feedback during key-presses. The proposed tracking algorithm is structured on two levels, namely a coarse level for foot regions, and a fine level for foot tips. Tracking works in real-time and handles efficiently feet regions merging/unmerging due to spatial proximity and cast shadows. The output of the tracking is used for the spatiotemporal detection of key-"press" events.

Key words: computer vision, motion tracking, real-time event detection

1 Introduction

The computer-mediated interaction between musician and instrument exhibits a significantly increased bandwidth of communication compared to the traditional musical performance act. As shown by Schloss and Jaffe [1], this new interaction paradigm goes beyond the one-on-one correspondence between the

* Corresponding author. Tel.: +1 418-656-2131 ext. 4786; fax: +1 418-656-3594.
Email addresses: fjean@gel.ulaval.ca (Frédéric Jean), aalbu@ece.uvic.ca (Alexandra Branzan Albu).

performer's actions and the sonic result, which is characteristic for acoustical instruments. Computer-mediated interaction defines meta-instruments, which use algorithms for music generation. While playing a meta-instrument, a particular gesture of the performer can have practically any musical result, which depends in fact on the software generating the music. Hence, there is no intrinsic relationship between the performer's actions and the sonic result.

The main issue raised by this fundamental change in the paradigm of interaction is related to the musician's perception of causal relationships when playing meta-instruments. In order to enable this perception, the one-on-one acoustical correspondence can be extended to a limited selection of sonic mappings of the performer's gesture. Specifically, the performer can choose among a limited number of mappings by controlling the software for music generation.

The technique proposed in this paper is generic allows for controlling the meta-instrument via foot gestures. Our work is generic and compatible with any type of foot-controlled meta-instruments. The current implementation of the proposed technique has used as musical meta-instrument the Mathews/Boie Radio Drum, which is a sensor able to track the position of two mallets in 3D and in real-time.

The musician controls the sound mapping of the Radio-Drum using a bank of organ-style foot pedals. The bank of organ-style foot pedals currently used with the Radio-Drum exhibits some significant usability limits. First, its limited portability and considerable weight makes it difficult to transport during concert-related travels. Second, the interaction is prone to errors caused by hitting two keys at the same time.

To address these limits, this paper describes a new prototype of keyboard, thereafter called *visual keyboard*, which functions using computer vision algorithms; the visual attribute shows the fact that, unlike a physical keyboard, our prototype does not involve any force feedback during key-presses. While a brief description of our study appeared in Jean et al. [2], the present paper contains a significant amount of new conceptual and algorithmic content with respect to [2]. The remainder of the paper is structured as follows. Section 2 describes related work in the area of vision-based perceptual interfaces, while section 3 presents our proposed approach. Section 4 shows our experimental results and their validation. Section 5 draws conclusions and describes future work.

2 Related Work

Since human perception is dominated by vision, humans are very skilled at visual methods of communication. A seminal work in social psychology by Mehrabian [3] shows that words contribute only with 7% to a message's meaning in face-to-face communication, while the tone of voice contributes with 38% and the body language contributes with 55%. Since the gestural component plays a major role in human-to-human communication, vision-based perceptual interfaces have partially translated this paradigm to human-computer interaction. The partial attribute comes from the fact that computer vision algorithms are currently able to recognize only a limited set of human gestures, as opposed to the human ability to detect a large variety of gestures, and also to differentiate among subtle nuances and extract meaning from the way a person performs a certain gesture.

Vision-based perceptual interfaces are based on algorithms for real-time tracking, modeling and recognition of human gestures from video input. Such interfaces have been developed for a variety of applications, ranging from health care (i.e. non-contact interfaces for the operating room [4]) to video games (*Eye-Toy* [5]) and to music generation and control. Tracking algorithms for vision-based perceptual interfaces can be classified into two main categories according to the motion modeling paradigm. The first category uses rigid appearance-based motion models and includes spatio-temporal template matching techniques (for instance Betke et al [6]) and blob-based tracking [7,8]. The second category uses deformable motion models or snakes in order to extract a representation of the bounding contour of the body part and to dynamically update it over time. For instance, Isard and Blake proposed in [9] and extension of the Condensation algorithm [10] which enabled them to develop a tracker able to follow the natural drawing action of a hand holding a pen.

The application context is essential for the process of selecting meaningful and easy-to-recognize gestures, as well as for determining the necessary level of accuracy and the maximal latency in gesture tracking and recognition. Therefore, the remainder of this section will focus only on perceptual interfaces for musical applications.

In 1919, Lon Theremin invented the world's first non-acoustic musical instrument based on proximity sensing; the Theremin sensed the distance to a performer's hands using changes in capacitance. Since then, technological advances have enabled the development of a wide diversity of interactive systems that perceive human motion and respond to it by affecting some aspect of the music generated or modified by computer. As shown by Winkler [11], such systems can be classified into two main categories. In the first category, human motion is used as direct input for generating music; in the second one, motion data controls musical parameters only. These two categories are non-exclusive, since some systems can both generate and control music using human motion information.

Systems that perceive human motion for music generation purposes are typically designed for entertainment or educational applications. Wren et al. [12] mention that such applications are based on a particular class of *hyperinstruments*, a term coined by Machover [13]. This specific class of *hyperinstruments* contains systems primarily designed for people with little musical background who still wish to participate in musical/multimedia creations. The computer that coordinates the music generation process provides the basic layer of musical knowledge necessary for the creation act. In addition to music, some systems produce related visual effects for a more immersive multimedia experience.

Creating music using human body motion is an intriguing idea, since it reverses the traditional causal relationship between music and dance. This concept has been explored in *EyesWeb* by Camurri et al. [14], and more recently in Castellano et al. [15]. In [14], human motion information is extracted using simple computer vision algorithms for the detection and tracking of the centers of mass of the silhouette and ten body joints; these algorithms use user-specified regions of interests and colour thresholds via a graphical user interface. In [15], the focus is on extracting whole body motion information rather than the motion of body parts.

DanceSpace, described in Wren et al. [12] and Sparacino et al. [7], is another perceptual space where dancers can generate music and graphics through motion. A performance in *DanceSpace* mimics the one of a street artist who has several instruments attached to her body parts. Compared to *EyesWeb*, the computer vision technology in *DanceSpace* is more sophisticated and mature; this technology, described in Wren et al. [8], uses a multi-class statistical model of color and shape for background subtraction and for tracking body parts in a wide range of viewing conditions.

Other applications targeting the creative participation of users with little musical background deal with interactive music rendition rather than generation. In Sim et al. [16], the style of the music performance (fast or slow, loud or soft) is controlled by the listener by interacting with the playback system via hand motions. Hand and finger motions are first tracked using a simple algorithm based on skin colour detection; next, information about the area of the moving body part (arm versus finger) and about the speed of motion is extracted.

The second category of music interfaces using human motion is focused on the control of musical parameters during live performance, rehearsal or replay. Such interfaces are mostly designed for professional musicians and for integrating prerecorded with live performances. The system proposed by Lee et al. [17] enables performers to freely manipulate the tempo of the recorded audio and video. Their approach first extracts the user's beats, and then maps these beats to the music rhythm by performing synchronized audio and video time stretching. Computer vision techniques are used during the user beat extraction process. The work of Behringer [18,19] deals with conducting digitally stored music by the visual tracking of the baton and the hand motion. This work has applications in conducting mixed ensembles of human musicians and electronic instruments. The visual tracking algorithm is based on extracting characteristics of conductor's hand trajectory during particular beats (i.e. a "three" beat).

Selecting the optimal body part(s) and their associated motion(s) for the control of music interfaces is a central design issue influencing the algorithm's computational complexity (and thus its ability to function in real-time), as well as the usability of the interface (i.e. how "natural" the interaction is). A comprehensive picture of the state of the art of computer vision techniques for large-scale body movement, gesture recognition, and gaze detection for multimodal human computer interaction is to be found in Jaimes and Sebe [20].

Some computer vision approaches for musical interfaces define their set of gestures among facial motions and/or expressions. The *Mouthesizer* proposed by Lyons et al. [21,22] extracts basic shape parameters of the mouth cavity such as height, width and aspect ratio and maps them to musical control parameters. The system described in Merril [23] processes infrared data for recognizing head nods, tilts, and shakes; these gestures are used for the discrete and continuous control of an effects unit.

One limitation of controlling musical interfaces via head or face movements is the extra cognitive load imposed on the performer. Nishikawa et al. [24] show that interacting via facial gestures is distracting and has a negative impact on the human performance when the task to be performed has a significant intrinsic cognitive load; this is also the case in live performance and rehearsal. Moreover, with regard to the live musical performance context, facial expressions play a central role in non-verbal communication and in connecting with the public. Therefore, controlling an interface with face or head gestures does not seem to be an appropriate solution for live concerts.

Other vision-based techniques for the control of musical interfaces track hand gestures. The system in Bradski and Davis [25] uses motion history gradients to recognize simple gestures and thus to control a vocal music synthesizer. For instance, waving gestures control the music tempo; waving with the left/right arm moves the music left/right. Hand gestures are not applicable for the control of a wide category of instruments, where hands are directly involved in playing the instrument.

The approach proposed in this paper belongs to the area of perceptual interfaces for the control of music production. We address the constraints imposed by the use of meta-instruments during live performance or rehearsal by tracking feet motion relatively to a visual keyboard. At the best of our knowledge, this is the first vision-based perceptual interface using feet gestures. The choice of this novel interaction paradigm is justified by application-specific user and task models. Indeed, the design of the visual keyboard is directly inspired from the traditional, organ-style bank of foot pedals. In the proposed prototype, colour cues and elevated height for the keys corresponding to black organ keys (see Figure 1(a)) are used to help performers locate the desired key. Moreover, a similar set of foot motions are used for controlling the interaction with the visual keyboard as in the traditional set-up. This similarity has a positive impact on the learnability and efficiency of the performer's interaction with the visual keyboard. The following section provides a detailed description of the proposed approach.

3 Proposed Approach

The structure of the proposed system is modular. The system is first *initialized* with user input about workspace and parameter values to be used by subsequent processes. Next, *background subtraction* is performed in order to find feet regions in the workspace. *Tracking* is necessary for establishing correct inter-frame correspondence between foot regions, and between foot tips. The output of the tracking module consists in spatiotemporal trajectories of labeled foot tips. These trajectories are used for the *temporal detection* and for the *spatial localization* of key-"press" events. Once a key-"press" event is temporally detected, its spatial localization is performed by identifying the key containing the contact point between the foot tip and the keyboard.

The system has real-time processing capabilities, since it is able to detect a key-"press" one frame (1/30 second) after its occurrence. Therefore, the entire sequence of steps with exception of initialization is performed on a frame-by-frame basis. The following subsections describe the algorithms involved in each of the modules of the system.

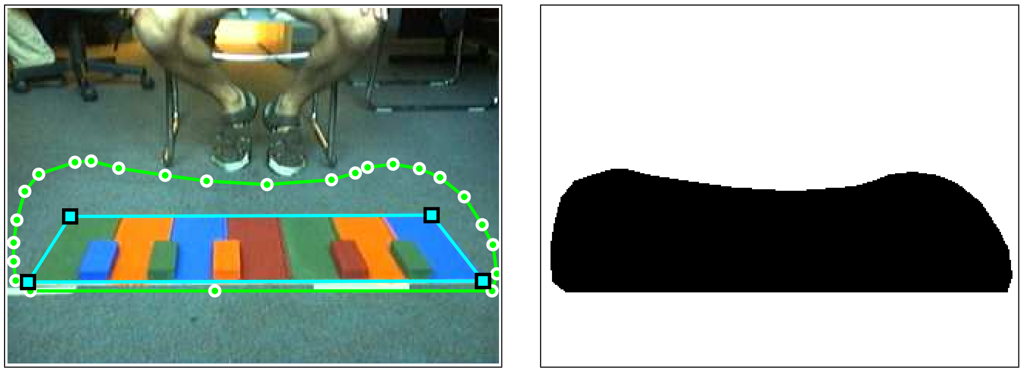

(a) (b)

Fig. 1. Defining the workspace and the keyboard corners' position during the initialization process. a) User-defined workspace boundary, specified as a discrete set of contour points (white circles); corners of the keyboard (black squares) are also user-specified. b) Binary mask W of the workspace; algorithms will only consider the region formed by black pixels.

3.1 Initialization

This process is performed only at the beginning of the musician-keyboard interaction. It assumes that both camera and keyboard have fixed positions during the musical performance. During initialization, the user specifies via a simple graphical interface the spatial location of the keyboard on the image plane as well as a "workspace" $W(x, y)$ surrounding this keyboard (see Figure 1). This workspace represents the region of the image where the feet and the foot tips will be tracked. The specification of a binary workspace mask is necessary for increasing the robustness of background subtraction and tracking. During initialization, the user also specifies all threshold values to be used in the further steps of the approach. The specification of these parameters is necessary in order to handle the variability in environmental conditions, as well as in the performer's key-"press" style. Details about these thresholds are to be found in the subsections describing the processes that are using them.

3.2 Background Subtraction

The algorithm used in this work computes statistics of the background when it is static, with feet not present in the workspace; for robust statistics, 2-3 seconds of video with static background only are sufficient. First- and second-order background statistics are computed as follows:

$$\begin{aligned} I_{B,\mu}(x,y,c) &= \frac{1}{T}\sum_{t=1}^{T} I_{B,t}(x,y,c)\ , \\ I_{B,\sigma^2}(x,y,c) &= \frac{1}{T-1}\sum_{t=1}^{T}\left[I_{B,t}(x,y,c) - I_{B,\mu}(x,y,c)\right]^2 \end{aligned} \tag{1}$$

where x and y define the spatial location of the pixel, c is the colour component ($c = 1$ for red; $c = 2$ for green; $c = 3$ for blue), T is the number of the considered static frames, and $I_{B,t}(x,y,c)$ is the colour image corresponding to frame index t. $I_{B,\mu}(x,y,c)$ and $I_{B,\sigma^2}(x,y,c)$ are the mean and variance images respectively. The background subtraction process uses these statistics for generating the foreground image $I_{M,t}(x,y)$ as follows:

$$I_{M,t}(x,y) = \begin{cases} 1 & \text{if } I_{\Delta,t}(x,y) \geq \tau_b \\ 0 & \text{otherwise} \end{cases} ; \tag{2}$$

$$I_{\Delta,t}(x,y) = \frac{\sum\limits_{c=1}^{3}\left[I_{B,t}(x,y,c) - I_{B,\mu}(x,y,c)\right]^2}{\sum\limits_{c=1}^{3} I_{B,\sigma^2}(x,y,c)} W(x,y) \tag{3}$$

where τ_b is a user-defined threshold in the initialization step. The use of a user-specified threshold allows for a correct background subtraction under various levels of environmental lighting. One may notice that the background subtraction process is based on the quadratic difference $I_{\Delta,t}(x,y)$, which conveys how different the intensity at location (x,y) is from the average background intensity. Figure 2 illustrates a typical output of a background subtraction process. Note that the result does not perfectly outline the feet location in the workspace, due to shadow, non-uniformities in the colour distribution of the feet etc.

3.3 Tracking

Tracking is performed at both a coarse level (foot) and a fine level (foot tip) on a frame-by-frame basis. This two-level tracking approach is able to follow in real-time the two main types of feet movements involved in operating a keyboard. Specifically, while placing the foot in front of the desired key in preparation for the key-"press" is a quasi-horizontal translatory motion, the actual key-"press" is a rotation in a plane orthogonal to the image plane. When a key is meant to be "pressed", the heel of the foot is placed on the ground in front of the key. The key-"press" is performed using a downward

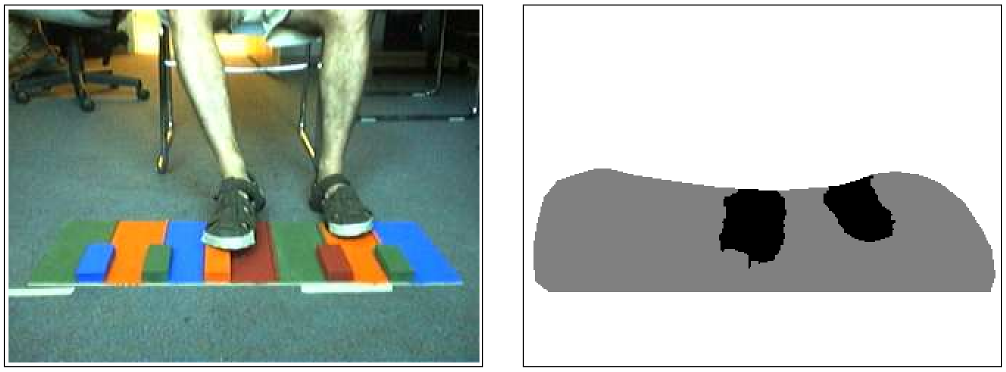


Fig. 2. a) Original frame with feet present in the workspace; b) Result of feet detection via background subtraction; $\tau_b = 8$; $T = 150$ static frames.

motion of the foot using the heel as pivot; thus, only the foot tip exhibits a vertical downward motion in the image plane.

One may note that, in order to detect key-"presses", only the downward motion of the foot tip needs to be detected. Coarse-level tracking provides search regions for the foot tips, thus it is a necessary preprocessing step for fine-level tracking. However, in some circumstances where foot regions merge (see section 3.3.4), fine-level tracking is performed first, and the resulting foot tip labels are propagated to foot regions.

The global flowchart of the tracking algorithm is shown in Figure 3. The following subsections describe approaches for foot region detection (coarse-level tracking), foot tip detection (fine-level tracking), as well as for inter-frame feet correspondence involved in both levels of tracking.

3.3.1 Detection of Feet Regions

To find the foot regions in the current frame t, all non-zero regions are first found on the foreground image $I_{M,t}(x, y)$ by performing a connected component analysis. The interaction with the keyboard at frame t may involve one foot, two feet, or no feet at all, if no interaction is present. Therefore, only the two largest regions in the foreground image are selected, if they exist. The number of foot regions under consideration at frame t, denoted by $N_R(t)$, can take three values: 0 (no region), 1 (one region), or 2 (two regions). For $N_R(t) = 2$, each region represents one foot. For $N_R(t) = 1$, the region can represent either one foot or both feet; the latter case occurs when feet are positioned close to each other so that their corresponding regions merge into one. Region merging is discussed in section 3.3.4. Each region $R^u(t)$, $u = 1 \ldots N_R(t)$ is represented by its rectangular bounding box; this representation will be further used for searching foot tips and for establishing inter-frame feet correspondence. Each bounding box is specified by its four corners given by their x and y coordi-

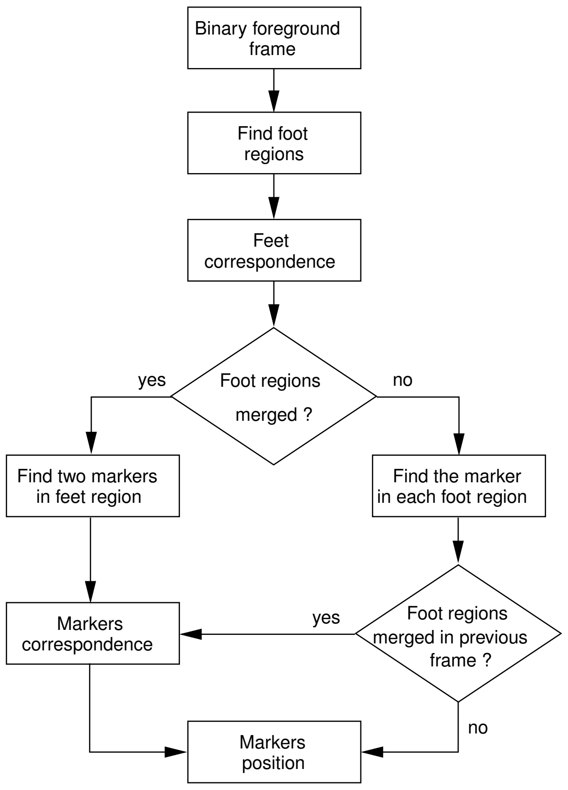


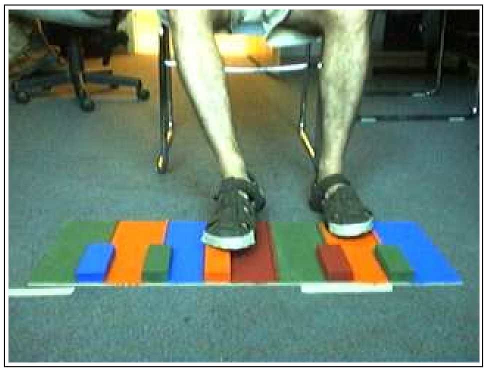
Fig. 3. Flowchart for tracking algorithm.

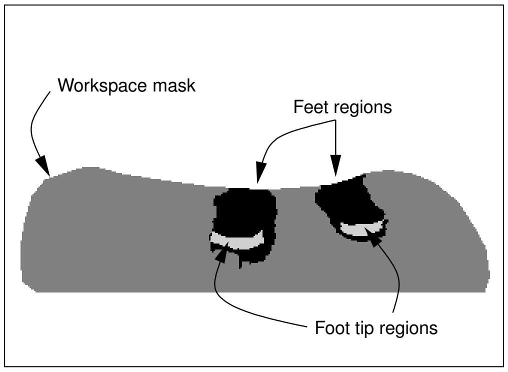
nates as follows: $R^u_{xL}(t)$ (left x), $R^u_{xR}(t)$ (right x), $R^u_{yT}(t)$ (top y), and $R^u_{yB}(t)$ (bottom y).

3.3.2 Detection of Foot Tips

To enable foot tip detection, the video sequences in the experimental database were acquired with shoes having their forward extremities outlined in white (see Figure 1(a)). Foot tip detection is therefore based on gray scale information; bright pixels (i.e. with a gray value greater than a user-specified threshold τ_g) in the foot regions are likely to belong to a tip. One should note that the brightness cue can be easily replaced with other cues such as colour, etc. and represents a convenient solution for outlining foot tips in a musical performance context.

For two disjoint foot regions, one tip region is detected inside each of these using connected component labeling. Specifically, in each foot region the largest connected component composed of bright pixels is associated to the foot tip. When region merging occurs, the two largest connected components composed of bright pixels are associated with foot tips. Figure 4 shows a typical example of detecting foot and foot tips.


(a)



(b)

Fig. 4. Example of foot region and foot tip detection. a) Original colour frame; b) Workspace mask, foot regions, foot tips; $\tau_b = 8$; $T = 150$ static frames; $\tau_g = 175$.

Foot tip regions are denoted $T^v(t)$, $v = 1 \ldots N_T(t)$ where $N_T(t)$ is the number of tips detected at frame t in a given foot region with possible values 0 (no foot tip), 1 (one foot tip), 2 (two foot tips). A foot tip region is represented by its rectangular bounding box given by the four coordinates of its corners as follows: $T^v_{xL}(t)$ (left x), $T^v_{xR}(t)$ (right x), $T^v_{yT}(t)$ (top y), $T^v_{yB}(t)$ (bottom y).

Two additional parameters are computed from the foot tip regions in order to enable temporal key-"press" detection (see section 3.4) and to identify the "pressed" key (see section 3.5). Specifically, the *foot tip position* is given by the coordinate vector of the mass center of the foot tip region, $\bar{\boldsymbol{p}}^v_{\mathrm{mc}}(t) = \left[p^v_{x,\mathrm{mc}}(t), p^v_{y,\mathrm{mc}}(t)\right]$, $v = 1 \ldots N_T(t)$. The *contact point* between the foot tip and the "pressed" key (in the eventuality of a key "press" event) is defined as $\bar{\boldsymbol{p}}^v_{\mathrm{contact}}(t) = \left[p^v_{x,\mathrm{mc}}(t), T^v_{yB}(t)\right]$, $v = 1 \ldots N_T(t)$. Thus, the x-coordinate of the *contact point* coincides with the x-coordinate of the mass center of the foot tip region, and the y-coordinate coincides with the bottom y of the tip's bounding box.

3.3.3 Establishing Feet and Feet Tips Correspondence

Feet correspondence is necessary for extracting the spatiotemporal trajectory of each feet tip; these trajectories will be further used for detecting key-"press" events. Feet correspondence is performed on a frame-by-frame basis by comparing the spatial location of feet bounding boxes in every pair of adjacent frames. This work uses two different techniques for feet correspondence, based on whether region merging between frames t and $t-1$ is present or not. The first technique is applicable when no region merging occurs, while the second handles region merging cases. Both techniques, as well as the procedure for detecting region merging are described below.

3.3.4 Detection Region Merging and Unmerging

Region merging between frames $t-1$ and t occurs when frame $t-1$ contains two foot regions ($N_R(t-1)=2$), and only one foot region is present in frame t ($N_R(t)=1$). In addition, the foot region in frame t overlaps partially with both foot regions in frame $t-1$, that is $R^1(t) \cap R^1(t-1) \neq \emptyset$ and $R^1(t) \cap R^2(t-1) \neq \emptyset$. Region unmerging is the inverse process of region merging. Specifically, region unmerging between frame t and $t-1$ occurs when frame t contains two foot regions, and only one foot region is present in frame $t-1$. The partial overlap condition remains the same as for region merging.

3.3.5 Simple Feet Correspondence

Simple feet correspondence is characterized by the absence of region merging between frames $t-1$ and t. For this case, feet correspondence is computed using as criterion the largest overlap ratio between the bounding boxes of the foot regions in frames t and $t-1$ respectively.

Let us consider $\vartheta(B_1, B_2)$, the overlap ratio between two bounding boxes B_1 and B_2. This ratio is defined as

$$\vartheta(B_1, B_2) = \frac{A(B_1 \cap B_2)}{\max\left(A(B_1), A(B_2)\right)} \tag{4}$$

where operator A returns the area of a bounding box, and $B_1 \cap B_2$ is the intersection of the two bounding boxes B_1 and B_2.

A 2×2 correspondence matrix C is formed, where each element c_{ij} is computed as follows:

$$c_{ij} = \begin{cases} \vartheta(R^i(t), R^j(t-1)) & \text{if } 0 \leq i \leq N_R(t),\ 0 \leq j \leq N_R(t-1),\ t > 1 \\ 0 & \text{otherwise} \end{cases} \tag{5}$$

where the bounding boxes $R^i(t)$, $R^j(t-1)$ belong to the i-th foot region in frame t, and to the j-th foot region in frame $t-1$ respectively.

The labeling process considers a set containing two labels, $\{l, \bar{l}\}$ namely. A complement operator functions over this set. Therefore, the two labels in the set are mutual complements, that is $l = \bar{\bar{l}}$. The purpose of labeling is not to distinguish between the left and right foot, but to maintain a correct inter-frame correspondence between feet regions in motion, and thus to enable the correct extraction of feet tip trajectories.

The initialization of the labeling process is random. When a foot region is first detected in frame t_0, and frame $t_0 - 1$ contains no feet regions, then the label of the detected foot region is randomly set to either l or $\bar{l}$. When two feet regions are first detected in frame t_0, each region is randomly assigned one different label.

Tracking feet over the frames succeeding t_0 implies labeling feet regions in correspondence over pairs of adjacent frames $(t, t-1)$. All regions $R^i(t)$ in frame t are labeled according to the inter-frame correspondence pattern $(t, t-1)$ as follows:

$$L(R^i(t)) = \begin{cases} L(R^{j^*}(t-1)) & \text{if } i = i^* \\ \overline{L(R^{j^*}(t-1))} & \text{otherwise} \end{cases}, \quad \text{with } [i^*, j^*] = \arg\max_{1 \le i,j \le 2}(c_{ij}) \tag{6}$$

where c_{ij} are elements of the correspondence matrix computed for frames $(t-1, t)$ with (5). Note that (6) holds only if at least one element in the correspondence matrix is non-zero. If this condition is not satisfied, then labeling is randomly reinitialized in frame t.

Let us consider the case where both frames t and $t-1$ contain two distinct feet regions, that is $N_R(t-1) = N_R(t) = 2$. The labeling process puts first in correspondence the regions the most likely to belong to the same foot in motion, namely the regions $R^{i^*}(t)$ and $R^{j^*}(t-1)$ whose overlap coefficient $c_{i^*j^*}$ in matrix C is maximum. Hence, region $R^{i^*}(t)$ receives the same label as region $R^{j^*}(t-1)$. The second region in frame t receives the complemented label, since it represents the other foot in motion.

Another simple correspondence case handled by (6) is the one where frame t contains 2 regions, while frame $t-1$ contains 1 region only. This case occurs when one foot is present in both frames t and $t-1$, while the second foot appears in frame t. In this case, the only non-zero element in matrix C is the maximum element $c_{i^*j^*}$. According to (6), the feet regions belonging to the foot present in both frames t and $t-1$ are labeled with the same label l , while the foot appearing in frame t receives the complemented label $\bar{l}$.

It is worth mentioning that the random initialization of labels does not allow for recovering information about the motion of one specific foot throughout successive appearances and disappearances from the workspace; each foot, either left or right, may receive in turn the same label over the course of a video sequence.

3.3.6 Foot and Foot Tips Correspondence in the Presence of Region Merging/Unmerging

The occurrence of region merging/unmerging between frames $t-1$ and t is detected as in 3.3.4. In the presence of region merging/unmerging, tracking is shifted at the feet tip level. The correspondence between frames $t-1$ and t is hence computed using the spatiotemporal overlap of feet tip bounding boxes. As for simple correspondence, the elements c'_{ij} of a 2×2 correspondence matrix C' are computed as follows:

$$c'_{ij} = \begin{cases} \vartheta(T^i(t), T^j(t-1)) & \text{if } 0 \leq i \leq N_T(t),\ 0 \leq j \leq N_T(t-1),\ t > 1 \\ 0 & \text{otherwise} \end{cases} \tag{7}$$

where bounding boxes $T^i(t), T^j(t-1)$ belong to the i-th foot tip in frame t, and to the j-th foot tip in frame $t-1$ respectively. The labeling process uses the same set of labels as in the simple correspondence case. The difference with respect to the simple correspondence is the fact that labels are assigned using the maximum overlap criterion applied to feet tip bounding boxes instead of feet regions. Therefore, labeling is computed sequentially by substituting c'ij to cij in (6). The formula using this substitution, thereafter called $(6')$, handles all the correspondence cases already discussed for (6), namely pairwise correspondence (see Figure 5(d)), as well as the detection of only one pair of foot tips in partial overlap (see Figure 5(e)).

In case of feet region merging, frame $t-1$ has two foot regions (each containing one foot tip respectively), while frame t has only one foot region containing two foot tips. Assuming that the foot regions in frame $t-1$ have already been labeled using simple correspondence, then their corresponding foot tips inherit the label of the foot region whom they belong to. Next, the two foot tips in frame t are labeled by evaluating their overlap with foot tips in frame $t-1$ as in $(6')$.

Conversely, in case of feet region unmerging, frame $t-1$ has one foot region containing two tips, while frame t has two foot regions (each containing one tip respectively). Since foot tips in frame $t-1$ have already received labels using $(t-2, t-1)$ inter-frame correspondence, foot tips in frame t are labeled with $(6')$. The labels of foot tips in frame t are then propagated to their corresponding foot regions.

There can be a number of frames between the occurrence of one region merging and its corresponding region unmerging. All these frames contain one foot region only with two foot tips. The labeling process is carried these frames at the foot tip level using $(6')$. Once region unmerging occurs, the foot tip labels

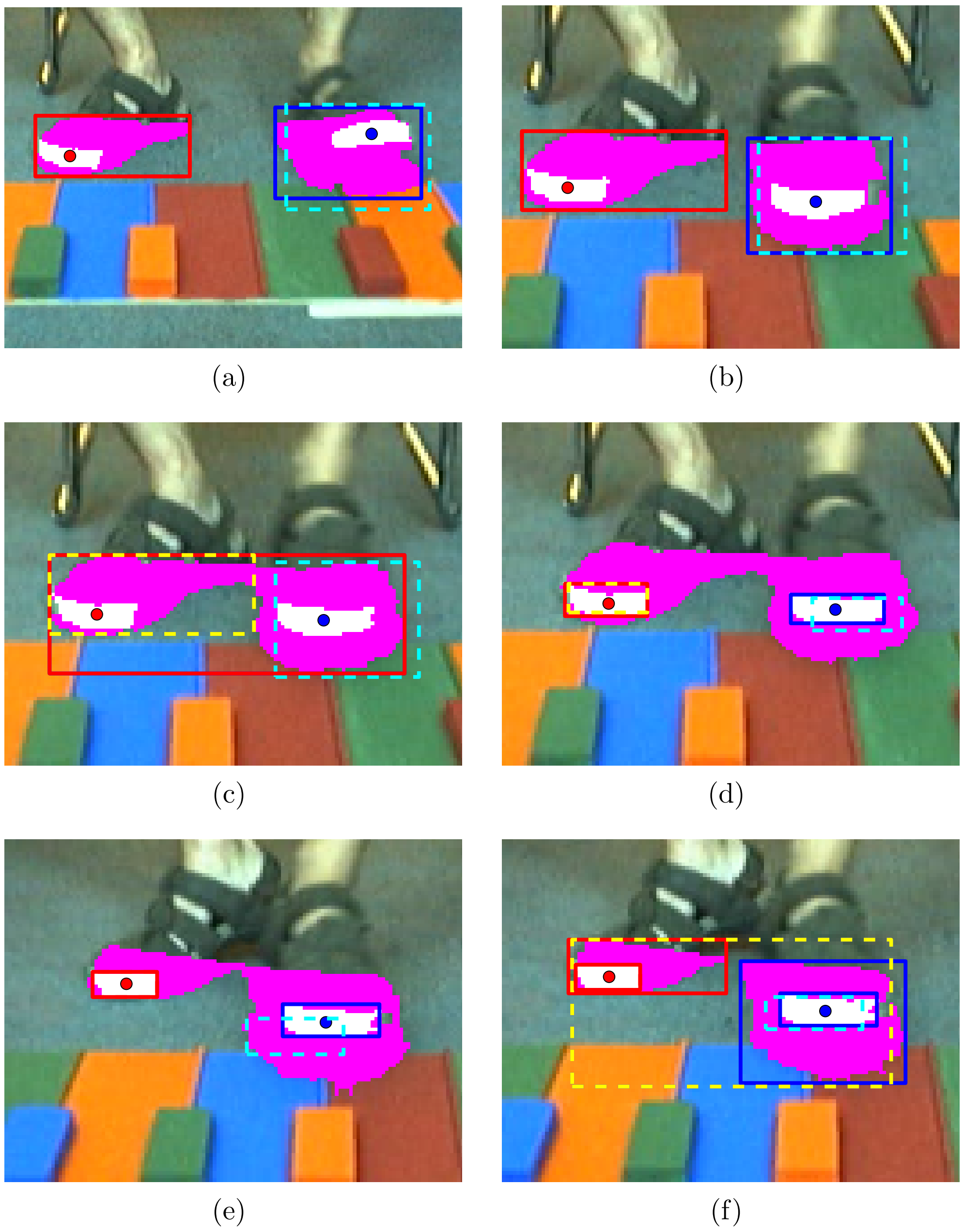


Fig. 5. Two-level tracking process over a sequence of 5 successive frames.

are propagated to their corresponding feet regions, and tracking at the foot region level is resumed.

Figure 5 illustrates an example of the two-level tracking process using a typical scenario of region merging-unmerging over a subsequence of successive frames. For clarity purposes, the frames in this subsequence are further referenced as 1 to 5. Bounding boxes (BB) in the current frame are shown with continuous borders, while those projected from the previous frame have dashed borders.

Figure 5(a) shows a simple feet correspondence computed between current frame (1) and its predecessor (0, not shown); both frames contain two distinct feet regions, and only one foot is in motion (overlapping BBs in shades of blue). Figure 5(b) shows the simple feet correspondence between current frame (2) and its predecessor (1); the viewpoint is slightly zoomed in with respect to Figure 5(a). Again, one may notice that one feet remains stationary (red BB), while the other continues its motion (overlapping blue BBs). As feet get closer to each other, region merging occurs in frame (3), which is the current frame for both Figure 5(c) and Figure 5(d). The detection of *region merging* is shown in Figure 5(d), which contains one red BB in current frame (3) in partial overlap with two dashed BBs projected from previous frame (2). In the presence of region merging, inter-frame correspondence is shifted at the foot tip level, as shown in Figure 5(e). Region merging persists in frame (4), which is the current frame for Figure 5(e). In frame (4), only one foot tip is in partial overlap with its correspondent for the previous frame (3); interframe correspondence is still correctly performed using the maximum partial overlap criterion and complemented labels. Figure 5(f) shows *region unmerging* in frame (5); labels are propagated from feet tips to their corresponding feet regions.

3.4 Temporal Detection of Key-"press" Events

As in the case of physical foot pedals, the velocity of the foot descending upon the key is maximal just before the hit occurs. Therefore, the keyboard hit detection is based on a maximum velocity criterion and is performed using the trajectory of the y coordinate of the foot tip's centre of mass. Key-"presses" performed with either foot are identified, since the spatiotemporal trajectories of feet tips are built using feet labels produced by the tracking process.

As specified in section 3.3.2, let $\bar{\boldsymbol{p}}_{\mathrm{mc}}(t)$ denote the projection onto the image plane of the trajectory of one foot tip tracked with the algorithm in 3.3. The y component of this trajectory is denoted by $p_{\mathrm{mc},y}(t)$. The dynamics of the foot tip motion, namely its velocity $v_y(t)$ and acceleration $a_y(t)$, are computed using first and second order differentiation. For that purpose, each foot tip in the current frame t must have been detected and labeled at frames t, $t-1$, $t-2$, $t-3$. The foot associated with that foot tip through the common label l is considered to have hit the keyboard at frame t_h if the following conditions are simultaneously met:

$$v_y^l(t_h-1) = p_{\mathrm{mc},y}^l(t_h-1) - p_{\mathrm{mc},y}^l(t_h-2) > \tau_v \tag{8}$$

$$\begin{aligned} a_y^l(t_h) &= v_y^l(t_h) - v_y^l(t_h-1) < 0 \\ &= p_{\mathrm{mc},y}^l(t_h) - 2p_{\mathrm{mc},y}^l(t_h-1) + p_{\mathrm{mc},y}^l(t_h-2) < 0 \end{aligned} \tag{9}$$

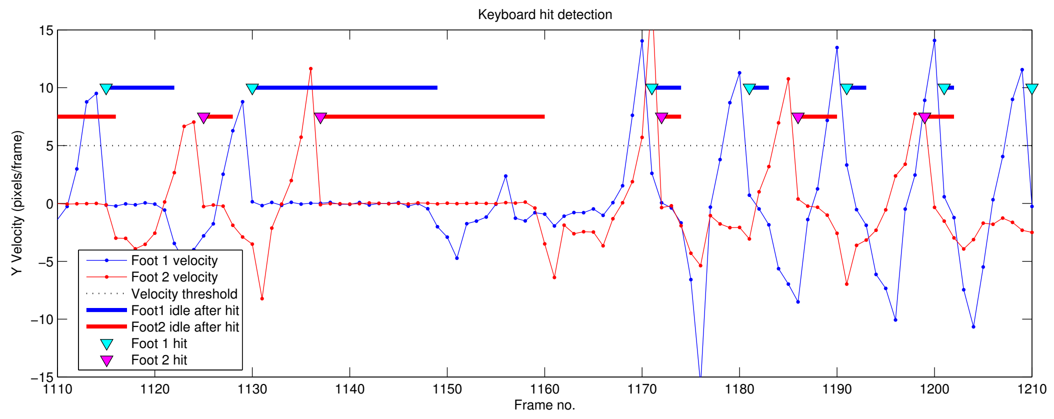


Fig. 6. Key-“press” and idle state detections. Keyboard hit times are shown by triangles; the following idle status is shown with a continuous line; τ_v=7 pixels/sec; τ_i=5 pixels.

$$\begin{aligned} a_y^l(t_h-1) &= v_y^l(t_h-1) - v_y^l(t_h-2) > 0 \\ &= p_{\mathrm{mc},y}^l(t_h-1) - 2p_{\mathrm{mc},y}^l(t_h-2) + p_{\mathrm{mc},y}^l(t_h-3) > 0 \end{aligned} \tag{10}$$

The above conditions associate a keyboard hit with the occurrence of a local maxima in the vertical velocity of the marker. To preserve only “significant” local maxima as opposed to those induced by noisy marker locations, the y-velocity in the previous frame $t_h - 1$ must be above a certain threshold τ_v. Our experiments have consistently used $\tau_v = 7$ pixels/second. The exact temporal location of the keyboard hit event is indicated by a zero-crossing (positive towards negative) in the vertical acceleration.

Key-“presses” can be events of short duration (user “presses” the key and immediately “releases” it), or they can last a longer time (foot remains in idle state on the key). The foot idle state indicates that the user has intentionally prolonged his key-“press” in order to continue the action associated with the “pressed” key. Once a key-“press” by foot l is detected at frame t_h, foot l is considered in idle state at frame t_c if the spatial location of its tip’s center of mass stays unchanged during $[t_h, t_c]$ as follows:

$$\left\| \boldsymbol{p}_{\mathrm{mc}}^l(t) - \boldsymbol{p}_{\mathrm{mc}}^l(t_h) \right\| \le \tau_i \quad \forall\, t \in [t_h, t_c] \tag{11}$$

where τ_i is a threshold (in pixels) for the maximum acceptable distance between the locations of the foot tip’s centre of mass. τ_i is typically set to 5 pixels. Figure 6 shows the vertical velocity of two feet for 100 frames, as well as the temporal occurrences of key-“press” events and idle states.

Once a key-"press" event is temporally detected at time t_h, its spatial location needs to be determined as well. Thus, it is necessary to find out which key has been "pressed", and thus to trigger the specific music software response associated to that key. The temporal detection of the key-"press" also locates the contact point between the foot and the keyboard, which belongs to the foot tip who "pressed" the key. As specified in section 3.3.2, this contact point is denoted as $\bar{\boldsymbol{p}}_{\text{contact}}(t_h)$; its coordinates are given in the image plane.

To identify the "pressed" key from the contact point, the region of the frame occupied by each key must be found. Since the location of keyboard corners in the image plane is known from manual initialization (section 3.1), a homography matrix $\boldsymbol{H}$ is built using the correspondence between the keyboard corners $\bar{\boldsymbol{b}}_i$ in the image and the real corners of the keyboard model $\bar{\boldsymbol{b}}_i^{\text{M}}$, $i = 1 \ldots 4$. The homography maps any point inside the keyboard model into a point in the image. Of particular interest for key identification are the corners of each key in the keyboard. The mapping of the known key corners in the model $\bar{\boldsymbol{k}}_s^{\text{M}}$ to their corresponding key corners in the image plane $\bar{\boldsymbol{k}}_s$, $s = 1 \ldots 30$ is computed as follows:

$$\hat{\boldsymbol{k}}_s = \boldsymbol{H}\hat{\boldsymbol{k}}_s^M \tag{12}$$

where $\hat{\boldsymbol{k}}_s^M = [\bar{\boldsymbol{k}}_s^{\text{M}},\ 1]^{\text{T}}$, $\hat{\boldsymbol{k}}_s = [e \cdot \bar{\boldsymbol{k}}_s,\ e]^{\text{T}}$ are expressed in homogeneous coordinates with e being the scale factor. The coordinates of the key corners in the image plane $\bar{\boldsymbol{k}}_s$ can be simply retrieved by dividing the x and y components of $\hat{\boldsymbol{k}}_s$ by e. These coordinates determine 12 distinct polygonal regions in the keyboard, one for every key in the image plane (see Figure 7(d)). Therefore, the key identification process finds the "pressed" key by determining the index of the polygonal region in the image plane which contains the contact point $\bar{\boldsymbol{p}}_{\text{contact}}(t_h)$.

4 Experimental Results

The keyboard prototype built for this work represents one octave of a piano keyboard and it is made of wood, glue, and paint. Its length and width-related dimensions are shown in Figure 7(b). In addition, small keys have a 2.5 cm height, which enables the user to differentiate easily between small and large keys. Video data was acquired with a Logitech® Quickcam® Pro 4000 with a 320×240 frame resolution and a frame rate of 30 fps. The acquisition process was performed under uncontrolled ambient lighting.

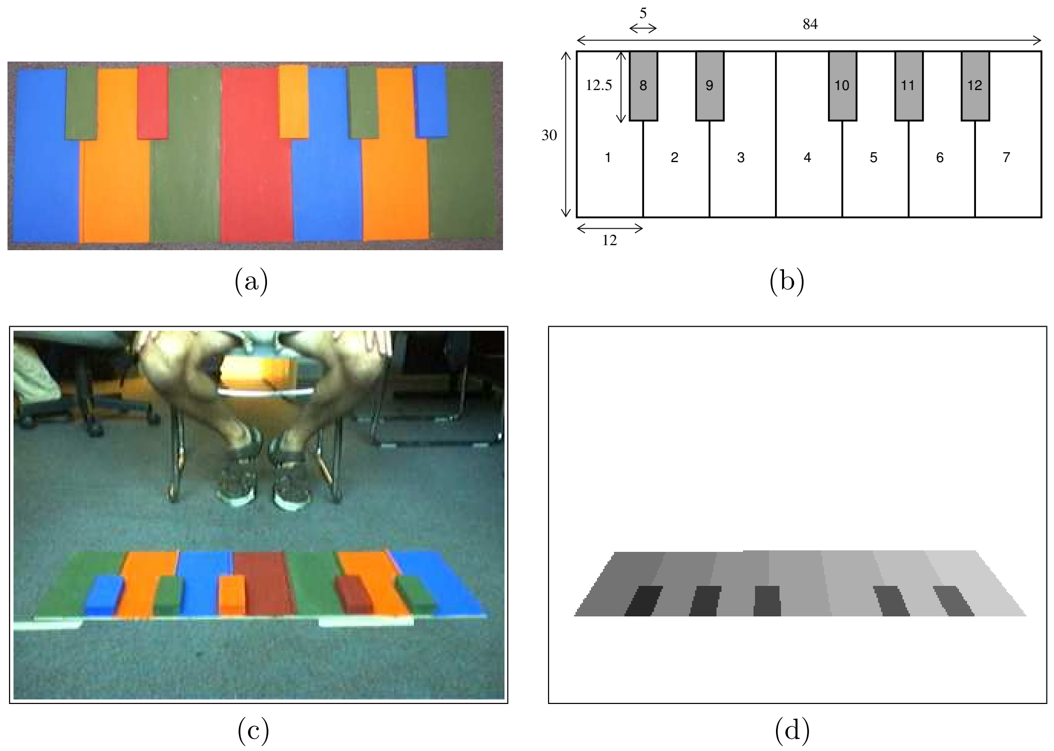


Fig. 7. a) Photographed keyboard; b) Keyboard model, with dimensions given in cm; c) Keyboard appearance in the image plane; d) Indexed polygonal regions in keyboard retrieved via homography of key corners.

The proposed approach was validated on 6 minutes (8307 frames) of video footage acquired with a music performer simulating foot actions which are typically used for the control of a meta-instrument with foot pedals. Specifically, video data contains slow and rapid foot motions, idle states, as well as simultaneous motion of both feet pressing on different keys. Table 4 summarizes the experimental validation of our approach.

The video footage was parsed into two sequences, acquired at different times and thus with a slightly different lighting. Based on the values shown in Table 4, the average ratio of missed key-“press” detections is 11.5%, while the average ratio of false key-“press” detections is 1%. The average error in key identification is 9.4%.

The main source of error for missed detections consists in the low values of speed at which the specific key-“presses” were performed. This issue may be addressed by learning the speed pattern of key-“presses” for a specific user from a training sequence.

The error in key identification occurs mostly for the elevated keys, which have a lower area than the keys located on the ground plane; also, the height of the elevated keys introduces a small error in the homography process (which assumes that the entire keyboard is planar). Future work will focus on mini-

Statistical performance measures	Sequence 1	Sequence 2
Total no. of frames	8307	1225
Total no. of detected key-press events	391	157
No. of key-press events detected correctly	328	145
No. of missed key-press detections	63	12
No. of false key-press detections	4	0
No. of errors in key identification	37	0

Table 1
Experimental results

mizing the key identification error by integrating information about the color of the “pressed” key with the approach described in 3.5. Color information can be obtained by automatic segmentation in the color space.

The proposed approach detects all key-“presses” one frame (i.e. 1/30 seconds) after their occurrence. Since all events are detected in real-time, our approach fulfils a critical requirement for a no-latency interaction between the visual keyboard, the MIDI software, and the musical meta-instrument.

5 Conclusion

This paper proposes a new system for interacting with a visual keyboard for the purpose of controlling music generated by meta-instruments. This system uses visual information only, with no force feedback, for the spatiotemporal detection of key-“press” events. As shown by our experiments, the system functions with high spatial accuracy and with very low temporal latency (1/30 seconds). Our proposed approach has significant advantages over the traditional marker-based tracking. Its spatio-temporal accuracy matches well the requirements of the application; there is no need for increasing the accuracy with optical marker-based technology. Moreover, optical-based marker systems are expensive, with low portability and thus difficult to deploy on stage.

From a practical standpoint, the approach described in this paper is, at the best of our knowledge, the first approach using feet motion for interacting with a perceptual interface. Tracking feet gestures represents an intuitive solution for the replacement of a physical organ-type bank of pedals with a visual keyboard. This solution enhances the learnability and ease-of-use of this new interaction paradigm. Other applications of human computer interaction based on feet gestures may be found for example in non-contact operating rooms (where hand gestures are typically used for performing the surgical intervention), virtual reality environments etc.

Our main theoretical contribution consists in the development of a two-level tracking algorithm which works in real-time and handles efficiently feet regions merging/unmerging due to spatial proximity and cast shadows. From a computational point of view, we believe that our tracking paradigm suits well the task at hand. As pointed by Isard and Blake in [9] , tracking algorithms based on deformable motion models would have difficulty meeting the real-time constraint, especially when dealing with the simultaneous motion of both feet and with uncontrolled environmental conditions.

Future work will focus on adapting our two-level tracking method for the real-time analysis of gestures performed with other body parts in motion. More specifically, we will investigate tracking of hand gestures, where global hand motion will be tracked at a coarse level and finger motion will be tracked at a fine level. Specific hand gestures such as pointing will enable the detection of hands and fingers based solely on skin colour and shape information.

References

[1] W. A. Schloss, D. Jaffe, Intelligent musical instruments: The future of musical performance or the demise of the performer?, Journal for New Music Research 22 (3) (1993) 183–193.

[2] F. Jean, A. B. Albu, W. A. Schloss, P. Driessen, Computer vision based interface for the control of meta-instruments, in: Proc. of 12th Int. Conf. on Human Computer Interaction (HCII), Beijing, 2007.

[3] A. Mehrabian, Silent Messages, California: Wadsworth Publishers Co., 1971.

[4] C. Graetzel, T. Fong, S. Grange, C. Baur, A non-contact mouse for surgeon-computer interaction, Technology and Health Care 12 (3) (2004) 245 – 257.

[5] Sony eyetoy.
URL `www.eyetoy.com`

[6] M. Betke, J. Gips, P. Flemming, The camera mouse: Visual tracking of body features to provide computer access for people with severe disabilities, IEEE Trans. on neural systems and rehabilitation eng. 10 (1) (2002) 1–9.

[7] F. Sparacino, G. Davenport, A. Pentland, Media in performance: Interactive spaces for dance, theater, circus, and museum exhibits, IBM Systems Journal 39 (3-4) (2000) 479–510.

[8] A. A. C. Wren and, T. Darrell, A. Pentland, Pfinder: Real time tracking of the human body, IEEE Trans. on Pattern Anal. Machine Intell. 19 (7) (1997) 780–785.

[9] M. A. Isard, A. Blake, A mixed-state condensation tracker with automatic model-switching, in: Proc. of IEEE Int. Conf. on Computer Vision (ICCV'98), 1998, pp. 107–112.

[10] M. A. Isard, A. Blake, Visual tracking by stochastic propagation of conditional density, in: Proc. of IEEE European Conf. on Computer Vision (ECCV'96), 1996, pp. 343–356.

[11] T. Winkler, Making motion musical: gesture mapping strategies for interactive computer music, in: International Computer Music Conference, 1995.

[12] C. Wren, S. Sparacino, A. Azarbayejani, T. Darrel, J. Davis, T. Starner, A. Kotani, C. Chao, M. Hlavac, K. Russel, A. Bobick, A. Pentland, Perceptive spaces for performance and entertainment: Untethered interaction using computer vision and audition, Applied Artificial Intelligence 11 (4) (1997) 267–284.

[13] T. Machover, Hyperinstruments: A Composer's approach to the Evolution of Intelligent Musical Instruments, Miller Freeman, 1992, pp. 67–76.

[14] A. Camurri, S. Hashimoto, M. Ricchetti, A. Ricci, K. Suzuki, R. Trocca, , G. Volpe, Eyesweb: Toward gesture and affect recognition in interactive dance and music systems, Computer Music Journal, MIT Press 24 (1) (2000) 57–69.

[15] G. Castellano, R. Bresin, A. Camurri, G. Volpe, Expressive control of music and visual media by full-body movement, in: Proc. of Conf. on New Interfaces for Musical Expression (NIME'07), 2007.

[16] T. Sim, D. Ng, Janakiraman, Vim: Vision for interactive music, in: Proc. of IEEE Workshop on Applications of Computer Vision (WACV), 2007.

[17] E. Lee, T. Karrer, J. Borchers, Toward a framework for interactive systems to conduct digital audio and video streams, Computer Music Journal, MIT Press 30 (1) (2006) 21–36.

[18] R. Behringer, Conducting digitally stored music by computer vision tracking, in: Proc. of the Int. Conf. on Automated Production of Cross Media Content for Multi-Channel Distrib. (AXMEDIS'05), 2005.

[19] R. Behringer, Gesture interaction for electronic music performance, in: Proc. of 12th Int. Conf. on Human Computer Interaction (HCII), Beijing, 2007.

[20] A. Jaimes, N. Sebe, Multimodal human computer interaction: A survey, in: IEEE Int. Workshop on Human Computer Interaction in conjunction with ICCV 2005, Beijing, 2005.

[21] M. J. Lyons, N. Testutani, Facing the music: a facial action controlled musical interface, in: Proc. ACM CHI, 2001.

[22] M. J. Lyons, M. Haehnel, , N. Tetsutani, Designing, playing, and performing with a vision-based mouth interface, in: Proc. of Conf. on New Interfaces for Musical Expression (NIME'03), 2003, pp. 116–121.

[23] D. Merril, Head-tracking for gestural and continuous control of parameterized audio effects, in: Proc. of Conf. on New Interfaces for Musical Expression (NIME'03), 2003, pp. 218–219.

[24] A. Nishikawa, T. Hosoi, K. Koara, D. Negoro, A. Hikita, S. Asano, H. Kakutani, F. Miyazaki, M. Sekimoto, M. Yasui, Y. Miyake, S. Takiguchi, M. Monden, Face mouse: a novel human-machine interface for controlling the position of a laparoscope, IEEE Trans. On Robotics and Automation 19 (15) (2003) 825–844.

[25] G. R. Bradski, J. Davis, Motion segmentation and pose recognition with motion history gradients, Machine vision and applications 13 (2002) 174–184.