UDC 7.05:004.9
ORIGINAL SCIENTIFIC PAPER
RECEIVED: 21-12-2010
ACCEPTED: 23-02-2011

ACTA GRAPHICA 187

Piano Crossing – Walking on a Keyboard

Authors

Bojan Kverh, Matevž Lipanje, Borut Batagelj, Franc Solina*

Faculty of Computer and Information Science,
Computer Vision Laboratory,
University of Ljubljana, Slovenia
**E-mail: franc.solina@fri.uni-lj.si*

Abstract:

Piano Crossing is an interactive art installation which turns a pedestrian crossing marked with white stripes into a piano keyboard so that pedestrians can generate music by walking over it. Matching tones are created when a pedestrian steps on a particular stripe or key. A digital camera is directed at the crossing from above. A special computer vision application was developed, which maps the stripes of the pedestrian crossing to piano keys and detects by means of an image over which key the center of gravity of each pedestrian is placed at any given moment. Black stripes represent the black piano keys. The application consists of two parts: (1) initialization, where the model of the abstract piano keyboard is mapped to the image of the pedestrian crossing, and (2) the detection of pedestrians at the crossing, so that musical tones can be generated according to their locations. The art installation Piano crossing was presented to the public for the first time during the 51st Jazz Festival in Ljubljana in July 2010.

Keywords:

Interactive Art Installation, Computer Vision, Background Removal, Music Tone Generation

1. Introduction

Computer vision is now widely used to sense the presence and actions of humans in the environment. Novel surveillance systems can reliably track and classify human activity, detect unusual events and learn and retrieve a number of biometric features *(Essa, 1999)*. Due to the low cost and the ubiquity of personal video technology, the research has recently shifted towards developing novel user interfaces that use vision as the primary input. In the area of personal computing, the most prominent areas of research are desktop interfaces that track gestures *(Kortum, 2008)*. On a wider scale, human motion can be used to interact with smart envi-

ronments *(Pentland, 1996)*, for example, trigger smart public displays *(Batagelj et al., 2008)*, or interact with virtual environments.

Real-time interaction of people with virtual environments is a well-established concept, but finding the right interface to do it is still a challenging task. Wearing different kinds of sensors attached to the body of the participants is often cumbersome. Computer vision offers the exciting possibility to get rid of such sensors and to record the body movements by means of a camera. On the other hand, people, their appearance (i.e. face), their emotions and their movements are increasingly becoming an important study object in computer vision research *(Essa, 1999)*.

The number of application areas for virtual environments has been growing ever since the costs of making virtual environments started going down. Sports games, used as training or for rehabilitation, are in general an attractive area for virtual technology. Many training machines or cycling, running, rowing are enhanced with a virtual world to make the training more interesting. Instead of a static scene in the fitness room, one can get a feeling of moving along a real scene or even racing against another real or virtual competitor. At their most complex, virtual exercisers are sophisticated simulations that deliver demands, stresses and sensations of a sport or exercise with unprecedented verisimilitude and precision.

Artists on the other hand experiment freely with new technologies and try to invent new and better ways of interfacing with virtual worlds *(Levin et al., 2002)*. More than ten years ago we have started the ArtNetLab *(ArtNetLab, 2010)* as permanent cooperation of the Computer Vision Laboratory at the Faculty of Computer and Information Science with the Academy of Fine Arts, both at the University of Ljubljana *(Solina, 2004b; 2000)*. This cooperation enabled the production of more than a hundred new media projects developed by students in the past ten years. Producing art installations gives us more freedom to experiment with the latest technology and to test, adapt or invent new methods in computer vision *(Peer & Batagelj, 2009)*. This also enables us to show our results to a wider public in an art gallery setting *(Solina, 2005)*. We initiated several art installations where computer vision played the central role. Some of the most successful ones are the following:

The installation *15 seconds of fame (Solina, 2004a; 2005)* was inspired by Andy Warhol's celebrated statement that "In the future everybody will be famous for 15 minutes" and his photography derived paintings of famous people. Our installation attempts to produce instant celebrities by reversing Warhol's process, making Warhol-like celebrity portraits of common people and hanging them on the walls of the gallery to make the them implicitly famous. Faces of people in front of the installation are detected from images taken by the camera, which is built into the picture frame where the portraits are displayed. One of the faces is randomly selected for the next 15 second celebrity and transformed in pop-art fashion using computer graphic methods.

The main goal of the *Smart Wall project (Peer and Batagelj, 2009)* is to provide a platform for rapid prototyping of computer-supported interactive presentations that sense human motion. The system is composed by a front end application, where the developer defines a number of hot spots in the camera view, a Hotspot Processor, which senses the activity in each of the hot spots, and a Player, which displays interactive content triggered by the activity in hot spots. By associating actions or sequences of actions in the environment to the actions in the interactive presentation, a variety of complex interactive scenarios can be developed and programmed very easily. Due to the modular architecture, the platform supports distributed interaction, connecting physical activity and content display at remote locations.

The installation *Virtual skiing (Solina, 2005; Solina et al., 2008)* is set up in a room with white walls and a floor covered with artificial snow. The skier stands on a pair of skis, which are attached to the floor. The virtual slope, as seen from the position of the skier, is projected on the entire wall in front of the skier. By using the same movements as on real snow the skier can negotiate the virtual slope as well. The movements of

the skier are captured by a video camera in front of the skier which in turn controls the animation of the virtual slope. The skier makes turns down the virtual slope to avoid scarcely planted trees just by changing the posture of his body. The interface is very intuitive since the skier just repeats the actions that he knows from real skiing, and learns to control his movements in the virtual world in less than a minute.

The *Virtual dance project (Dovgan et al., 2008)* provides a flexible framework which allows a dancer to set up an interactive virtual dance performance by defining markers, videos and interactive visual icons associated with markers. The system is then able to interact between dancer's real movements and his virtual movements. We used standard tracking methods and modified them to support fast moving markers, small markers and discontinuous tracking of markers. The real dance and its virtual presentation are inseparably connected because of the real time video processing. Every movement in the real world immediately produces a movement in the virtual world. Dancers can observe the virtual dance that is produced by their movement as a video projection. The dancers can therefore interact with the virtual space through their dance.

A classical or static anamorphic image requires a specific, usually a highly oblique view direction, from which the observer can see the anamorphosis in its correct form. Dynamic anamorphosis *(Solina & Batagelj, 2007)* adapts itself to the changing position of the observer so that wherever the observer moves, he sees the same undeformed image. The dynamic changing of anamorphic deformation, in concert with the movement of the observer, requires the system to track the 3D position of the observer's head and the recomputation of the anamorphic deformation in real time. This is achieved by using computer vision methods which include face detection and tracking of the selected observer in 3D. We used dynamic anamorphosis for the first time in the context of an art installation. A human face staring directly ahead is projected on the wall of a dark room, so that the only visible cues seen by the user are given by the projected image. The position of a single user determines the anamorphic deformation, so that the user always sees the same, respectively undeformed image from all positions in the room. Dynamic anamorphosis therefore disassociates the geometric space in which the user moves away from the visual cues they see, since wherever the observer moves, they see the same image. Henceforth there is no way to escape the gaze of the projected face in this art installation. On a symbolic level, the installation epitomizes the personification of ubiquitous video Surveillance systems *(Levin et al., 2002)*.

Piano crossing, the subject of this article, is an art installation which turns a pedestrian crossing into a piano keyboard, where pedestrians themselves generate music by walking over it. The installation was initially conceived by Matevž Lipanje for his master's thesis *(Lipanje, 2010)*. It consists of a pedestrian crossing with added black lines, computer-equipped with loudspeakers and a digital camera pointed to the pedestrian crossing from above. A specially developed computer vision application maps piano keys to the stripes of the pedestrian crossing and then detects which stripe a pedestrian is located upon at a particular moment, so that a corresponding music tone can be generated. Each white line of the crossing represents one white key on the piano, while black lines are laid down between white lines to represent the black keys. A pedestrian who walks over the crossing makes music with his feet similarly just as a pianist does with his fingers. In fact, merely the "centre" of a pedestrian and its relation to the keys must be detected.

In this article we present technical details of the installation, especially the computer vision methods for segmenting the pedestrian crossing into an abstract keyboard and detecting the pedestrians and their position at that crossing.

The idea of playing the piano with feet instead of hands is in the air for quite some time. The Philadelphia engineer, kinetic artist and inventor Remo Saraceni created the Big piano or Walking piano in 1976 *(Big piano, 2010; Saraceni, 2010)*. This piano can be played by actually stepping on the piano keys. It was featured in films and is today installed at several public

institutions, such as art centres, hospitals, toy stores and shopping malls.

The Walking piano is produced in several versions, small for children and bigger for adults. The main motivation in producing the Big piano was entertainment.

A slightly different motivation lies behind the Piano stairs which are installed in the Stockholm underground station *(Piano stairs, 2009)*. Individual stairs were turned into piano keys by mounting appropriate touch sensors on them. The piano stairs should motivate people to climb the stairs instead of riding on a moving stairway.

The visual similarity between pedestrian crossings and keyboards was not lost out with visual designers *(Design crosswalks, 2010; Walk the tune, 2010)*. At some of those crossings piano music soundtrack starts playing when a pedestrian steps on the crossing. Most of them just serve for various promotional purposes.

The main purpose of our interactive piano crossing is promotion with an added twist of interactivity. The installation does not require any physical change of the road surface since the location and time that a person touches the stripes of the zero crossing is detected by computer vision methods.

The rest of the paper is organized as follows: Section 2 describes the generation of an abstract keyboard (nicknamed *zeboard*), in Section 3 two methods for background modelling are presented, while Sections 4 and 5 are reserved for experimental results and conclusions.

2. Generating the zeboard

With this initial operation, it is necessary to transform the image of the crossing into a *zeboard* (an abstract keyboard made out of a pedestrian crossing) by segmenting it into areas of white lines and spaces between them. The process should be independent of the length and width of the crossing, which also makes the placing of the camera unbound to some exact position and not vulnerable to changing illumination conditions. The process should also be as automatic as possible. Each key on the *zeboard* must be associated with an image region, MIDI number and key status, whether it is pressed or not. This allows the next stage, Locate and play to actually generate musical tones as a pedestrian walks over the crossing. MIDI (Musical Instrument Digital Interface) is a protocol that transmits information such as the pitch and intensity of musical notes to play.

The process of generating the *zeboard* is divided into the following steps:

1. image pre-processing:
   - noise reduction,
   - transforming the colour image into intensity image,
2. pyramid segmentation,
3. search for the contours of white and black keys:
   - thresholding,
   - contour analysis,
4. generation of a MIDI keyboard.

2.1. Image pre-processing

Within this stage, we first reduce the noise in the captured image. Since the Gaussian noise is most likely to be present, we use a Gaussian filter to convolve our image with. We used a Gaussian filter in the form of 3 x 3 matrix. After noise reduction, the image is transformed into intensity image using the following formula:

$$C = 0.299R + 0.587G + 0.114B \qquad (1)$$

Parameters R, G and B are intensities of red, green and blue components for each pixel in our original image, while C is the resulting intensity of the same pixel in the intensity image. *Fig. 1* shows the results of pre-processing.

2.2. Pyramid segmentation

Pyramid segmentation is used to divide the intensity image into coherent regions. Ideally, each region should represent a key on the *zeboard*. The process is divided into the following steps:

1. generation of a Gaussian pyramid,
2. pixel linking,
3. grouping linked pixels into regions.

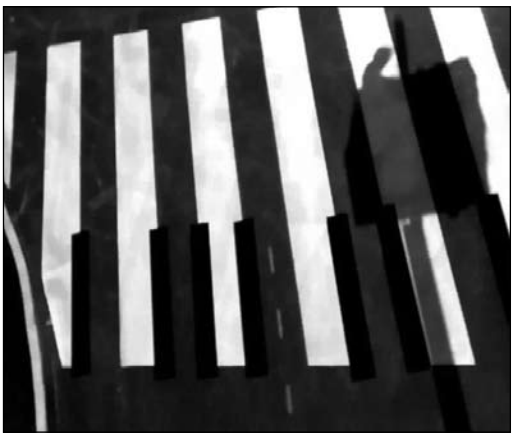

Figure 1. During pre-processing the original colour image is converted to an intensity image.

Steps 2 and 3 are repeated until the desired level of segmentation is reached.

A Gaussian pyramid is a collection of images, generated from the original image by downsampling. The original image is downsampled until we reach the desired resolution. To obtain the image on level i+1 we first convolve the image on level i with a Gaussian filter and then leave out pixels in even rows and even columns. Segmentation starts at the highest level of the pyramid, i.e. at the image with the lowest resolution. Relations of inheritance are formed between consecutive levels, as each pixel on level i + 1 has four potential ancestors on level i.

The relation between pixel a on level i and pixel b on level i+1 is formed if $p(c(a), c(b)) < t_1$, where c(a) is the intensity level of pixel a, t_1 is a threshold value and function p is defined as:

$$p(c_1, c_2) = |c_1 - c_2| \qquad (2)$$

Relations obtained in this way form linked pixels which are then grouped into regions. Two areas of linked pixels A and B belong to the same group if $p(c(A), c(B)) < t_2$, where function p is the same as before, t_2 is another threshold value, while c(A) is the average intensity for the entire area A.

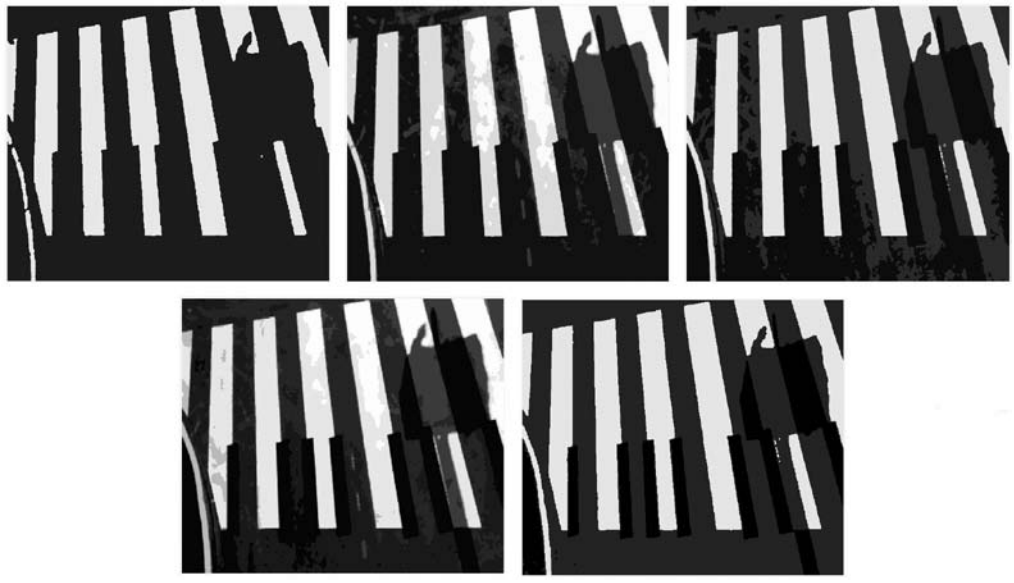

Figure 2. Results of pyramid segmentation for different threshold values t_1 and t2, from the worst (top left) to the best (bottom right).

The optimal threshold values t_1 and t_2 are obtained empirically. The user interface of the application provides two sliders, which allow us to change those values and see which the most appropriate ones are. *Fig. 2* shows the results of segmentation for different threshold values.

2.3. Search for the contours

The first part of this step is thresholding the intensity image into a binary image. We need two different binary images, one containing black lines on white background and another, containing only white lines on black background, as shown in *Fig. 3*.

We can search for the contours around areas on the binary image, which represent the foreground. Contours can be represented either by a list of pixels, located on the border of the area, or by a list of vertices of the polygon that surrounds the area. We use the latter and we want our polygon to be as simple as possible to facilitate further analysis. Thus, starting with a polygon represented by all of the pixels and located on the border, we use the Douglas-Peucker algorithm to minimize the number of polygon vertices (*Douglas and Peucker, 1973*). Since there can be several disturbances on the image of the pedestrian crossing such as shadows, also to be seen on the example images (*Figs. 1, 2, 3 and 4*), we needed to set some criteria, by which the individual areas are accepted or not accepted as *zeboard* keys. The areas whose width and height do not match the predefined values are filtered out. The result of searching for contours is shown in *Fig. 4*.

2.4. Generation of a midi keyboard

In the final step we actually generate necessary instances of data structures. Data structure, representing an individual key, consists of the following:

- contour representation, using OpenCV library structure CvSeq,
- the MIDI number of the key,
- a flag, which denotes whether the key is pressed or not,
- pointer to the next key.

Data structure, representing the entire keyboard, is very simple and consists of the pointers to the list of white and black keys, respectively.

3. Locate and play

In order to generate a MIDI sequence in accordance to the pedestrian walking over the crossing, we need a way to distinguish a pedestrian from the background in a video, obtained by our camera. Thus we need a background model which is then subtracted from the processed image. This model needs to be regularly updated during the locating procedure, which enables it to adapt to the illumination and other changes of the scene and effectively detect the pedestrians in the foreground. The application makes use of two adaptable techniques for separating foreground from the background: simple and advanced technique.

Figure 3. Thresholding the intensity image (left) into two binary images, with black lines on white background (centre) and white lines on black background (right).

3.1. Separation of foreground from background - simple technique

In this case, the first frame of the video is regarded as the reference background image. We can denote this frame as B. It is important that there are no pedestrians, vehicles or other objects on the pedestrian crossing when we start recording the video. At time t, a frame I_t is obtained from the camera and we calculate the difference $||I_t - B||$. Since I_t and B are both intensity images, the difference between the two is calculated as a sum of absolute differences between intensity values of pixels from the same location on both images.

If the difference exceeds some predefined threshold value T, the frame I_t contains objects in the foreground. T should denote the maximum level of noise in the video frames and it was set to 20 in our experiments. This approach works well in the areas with constant illumination, which is not the case on open scenes under natural illumination. For outside scenes the background model has to adapt to the changes of illumination.

We used the following formula to obtain this:

$$B_t = (1-\alpha)B_{t-1} + \alpha I_t \qquad (3)$$

This formula was first used in the system PFINDER *(Wren et al., 1997)*.

The speed of adaptation to the new incoming frames is regulated by parameter α. We discovered that the optimal value for our application is 0.003. With this formula each new frame is slightly integrated into the reference background image B, which consequently adapts to the slow illumination changes. Note that every pixel from the current frame contributes equally to the adaptation, regardless of whether it belongs to the background or the foreground. Slow moving objects or pedestrians could be integrated into the background image entirely over time. To avoid such problems, we could use a much smaller value of α (or even 0) for pixels which belong to the foreground. The results of using this technique are shown in Fig 5.

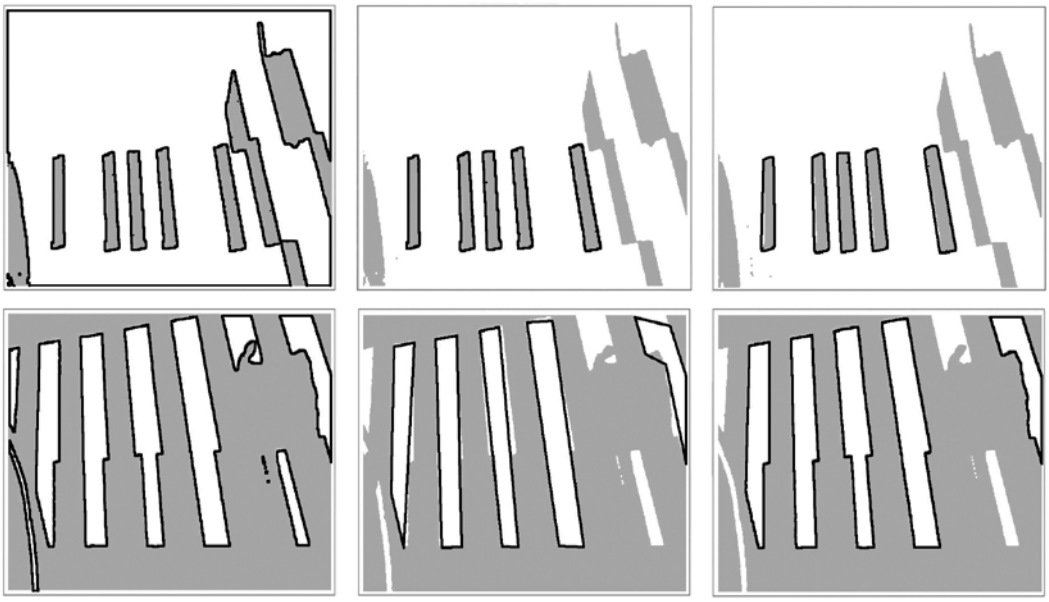

Figure 4. Searching for contours, representing black (top) and white (bottom) keys. Initial contours (left) are filtered using the compliance to their widthand height (middle) and simple polygon contours are calculated from the remaining ones (right).

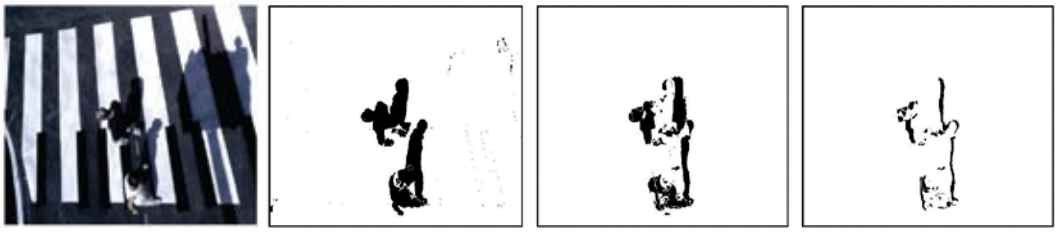

Figure 5. Simple technique of separating the foreground (two pedestrians and their shadows) from the background with different α values. From the left: original image, difference images with values α = 0.003, α = 0.1, α = 0.5. If the value of α is too large, only the silhouette of the foreground is detected.

This technique works well if we have a static background and fast moving objects in the foreground. However, when the objects in the foreground move too slowly, they get integrated into the background reference image, making the detection more difficult and inaccurate. We thus had to find a better solution for background removal.

3.2. Separation of foreground from background – advanced technique

The most popular method for background modelling, MOG *(Stauffer and Grimson, 1999)*, where each pixel is modelled as a combination of K Gauss distributions, proved to be too slow to use in real time with video of size 640 x 480. We therefore decided to use a technique where the image is modelled with the help of a codebook *(Kim et al., 2005)*. The principle of encoding image changes in the given time frame functions as follows: the values of each pixel element which change during the time period that is modelled are encoded with a set of code words or a codebook, which represents the compressed model of the background for this pixel location over this time period. The method is appropriate for static and dynamic environments.

Each pixel in the image is represented by its codebook C = $\{c_1, c_2, \ldots c_L\}$, where $c_1, c_2, \ldots c_L$ are code words and L is the number of code words for a specific pixel location. Normally, L << N, where N is the number of video frames or images in the sequence that is modelled by the codebook. A codebook C for a given pixel location is generated from the time series of values of that pixel X = $\{x_1, x_2, \ldots x_N\}$ over N video frames, where x_i is the RGB vector of the pixel in the i-th frame. Each code word c_j is represented by two vectors:

$$v_j = \left(\overline{R}_j, \overline{G}_j, \overline{B}_j\right)$$
$$aux_j = \left(\breve{I}_j, \hat{I}_j, f_j, \lambda_j, p_j, q_j\right) \qquad (4)$$

These parameters are explained below

- v_j — the average RGB colour of the code word,
- $\breve{I}_j, \hat{I}_j$ — minimal and maximal brightness,
- f_j — codeword frequency,
- λ_j — longest interval of absence of this code word,
- p_j, q_j — first and last appearance of the code word.

The algorithm which generates the codebook from the time series X for each pixel location is quite straightforward. For each pixel vector x_i in the time series it searches for a compatible codeword in C. The RGB value of x_i = (R, G, B) is compatible with codeword c_j if the following two criteria are both true:

$$colordist\left(x_i, c_j\right) \leq \in_1$$
$$\breve{I}_j \leq brightness\left(x_i\right) \leq \frac{\hat{I}_j}{\alpha} \qquad (5)$$

Function brightness simply calculates a sum of pixel RGB components, while colordist calculates the distance between the pixel colour and the line that represents the colour regardless of the brightness in RGB space. This line goes through the origin of RGB space and the average colour of the code word as shown in *Fig. 6*. It is defined as follows:

$$colordist(x_i, c_j) = \sqrt{\frac{\|x_i\|^2 - (x_i \cdot c_j)^2}{\|c_j\|^2}}$$

$$\|x_i\|^2 = R^2 + G^2 + B^2 \qquad (6)$$

$$\|c_j\|^2 = \overline{R}_j^{\,2} + \overline{G}_j^{\,2} + \overline{B}_j^{\,2}$$

$$(x_i \cdot c_j)^2 = R \cdot \overline{R}_j^{\,2} + G \cdot \overline{G}_j^{\,2} + B \cdot \overline{B}_j^{\,2}$$

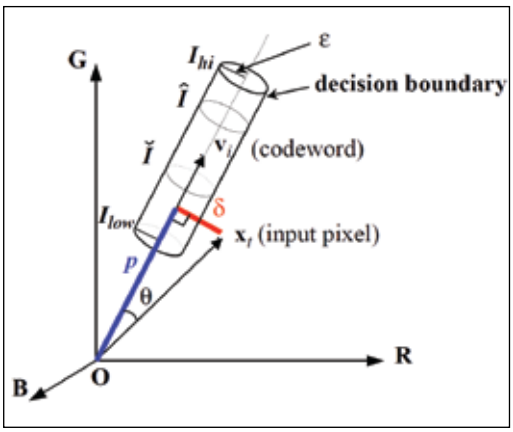

Figure 6. Distance between pixel and code word in RGB space is drawn in red. Note that xxx and xxx .

Parameter α should be set between 0.4 and 0.7. If we find a compatible code word (let denote it with index m), we update it as follows:

$$v_m \leftarrow \left(\frac{f_m \overline{R}_m + R}{f_m + 1}, \frac{f_m \overline{G}_m + G}{f_m + 1}, \frac{f_m \overline{B}_m + B}{f_m + 1} \right) \qquad (7)$$

$$\check{I}_m \leftarrow \left(\check{I}_m, brightness(x_i) \right)$$

$$\hat{I}_m \leftarrow \left(\hat{I}_m, brightness(x_i) \right)$$

$$f_m \leftarrow f_m + 1$$

$$\lambda_m \leftarrow max(\lambda_m, i - q_m)$$

$$q_m \leftarrow i$$

If we cannot find for a compatible code word, we increase by one, create a new code word and set its parameters as follows:

$$v_L \leftarrow x_i \qquad (8)$$

$$\check{I}_L \leftarrow min\left(\check{I}_m, brightness(x_i) \right)$$

$$\hat{I}_L \leftarrow max\left(\hat{I}_m, brightness(x_i) \right)$$

$$f_L \leftarrow 1$$

$$\lambda_L \leftarrow i - 1$$

$$p_L, q_L \leftarrow t$$

After all in the time series X have been evaluated and the codebook C has been created for a given pixel location, we check each codeword c_j for an eventual update of λ_j which denotes the longest interval of absence of this codeword. If we consider the series of N pixel values X a circular list and if the number of frames between the first appearance (p) and the last appearance (q) of codeword c_j in it is greater than current λ_j , then λ_j is updated to that number of frames, which is $N - q_j + p_j - 1$.

Finally, we need to scan the codebook and filter out the code words, where λ is larger than some threshold value T_M. This rejects code words, which have not appeared in the image for a long time and thus most likely belong to some foreground objects. In our application, T_M has been set to N/2.

To determine whether a given pixel in a new video frame belongs to the foreground or the background, all we have to do is check whether the value of this pixel is compatible with any of the code words for this pixel location.

Compatibility is checked by the same set of conditions as in *Eq. 5*. If a pixel in the new frame is compatible with at least one of the code words for this pixel location, it is labelled as background. Some experimental results using this technique are shown in *Fig. 7*.

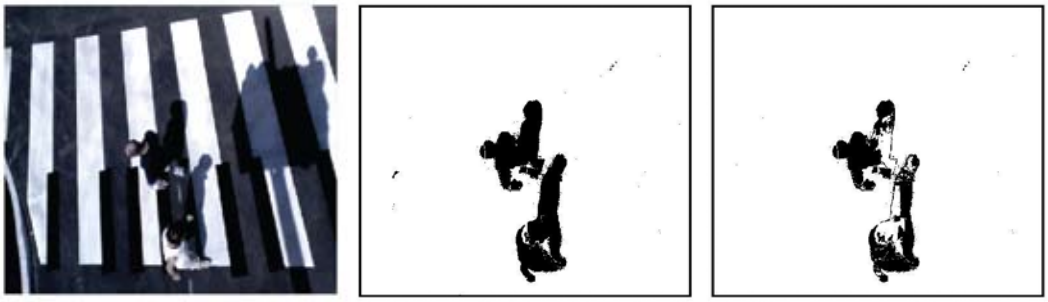

Figure 7. Codebook technique of separating foreground (two pedestrians and their shadows) from background in images. The middle and right image show the results with and without the filtration of code words with large λ, respectively.

Note that it is also possible to use this code word method with other colour systems than the RGB, for example with the YUV or HSV colour systems, where one of the components is brightness itself. The main reason for using other colour systems is that most of the changes between video frames are due to a change in brightness and not in the colour itself. When one of the colour system components is brightness, we do not need anymore the colourdist function to determine the compatibility between a pixel and a code word. Instead, we can simply check if a value of each pixel's component is within the code word interval, like we did with the function brightness in the RGB colour model (*Eq. 5*). This way, the entire process of separating the foreground from the background in the image becomes simpler and faster, and this is very important for real-time applications like ours.

In our experiments, the time series of N = 100 video frames were used to build the initial codebook which was then updated after every 500 frames.

3.3. Locating the pedestrians and generation of MIDI sequence

The result of separating the foreground from the background is a simple binary image, on which pixels that belong to the foreground objects are marked by white pixels. The next step is to group the pixels belonging to a single pedestrian, and to calculate their mass centre. An iterative algorithm is used for region labelling (*Rosenfeld and Pfaltz, 1966*), which uses an array of labels. The method scans the binary image and labels every white pixel encountered with the label of one of its white neighbours. If such a neighbour does not exist, a new label is created and assigned to the pixel. In the next step, the array of labels is scanned in search of equivalent labels, i.e. those which are assigned to neighbouring pixels. Pixels labelled with equivalent labels are considered to be part of a single region.

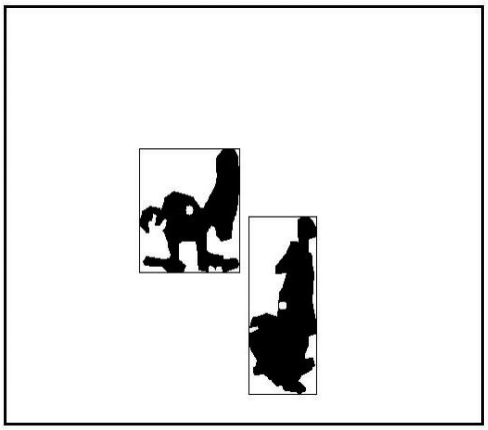

Figure 8. Labelled regions in the binary image showing the foreground objects (two pedestrians and their shadows) and their corresponding frames.

After region labelling, we compute the frame around each region (*Fig. 8*) and its mass centre, which is then regarded as the position of a corresponding pedestrian. If the camera is not positioned above the pedestrian crossing, but looks at it from the side, the calculated mass centre could not appear on the line which the

pedestrian is moving along at the moment. If a pedestrian projects a shadow, the shadow also moves with the pedestrian and is integrated into the region whose mass centre we compute. In such cases the mass centre shifts towards the feet of the pedestrian. In general, if the viewing angle of the camera differs considerably from the vertical orientation, we have to translate the calculated mass centre for some predefined vector, which could be determined empirically.

The generation of the final MIDI sequence is straightforward; the application calculates the location of the pedestrian at each given video frame and compares it with the initially detected *zeboard* key contours. If the pedestrian's mass centre is inside a certain key contour, the corresponding MIDI number for this key is added to the MIDI sequence. The algorithm for generating the MIDI sequence runs for every image in the image sequence over all keys on the keyboard to check if any of the detected blobs corresponds to it.

The key is also labelled as active, which prevents the generation of several activations of the same tone when the pedestrian is crossing a single *zeboard* key area. Only after the pedestrian leaves the key area, the key is labelled as inactive again. This method corresponds to the mechanics of a real piano; once a key is pressed, it cannot be pressed again before it is released. If several people are walking on the *zeboard* at the same time, several tones are generated, each corresponding to the mass centre of individual blobs. If blobs of several people temporarily merge, just a single tone is generated according to the mass centre of the merged blob. Once a blob separates again into two blobs, each of the two starts generating their own tones.

4. Experimental results

We tested our application on three pedestrian crossings. The first one was a smaller version of a pedestrian crossing, set up inside a building; the second was a real pedestrian crossing at a newly built crossroad which has not yet been open for traffic. The third one was just a minimized version made out of paperboard which was used to test different levels of illumination. We achieved good results in detecting white and black lines on pedestrian crossings and locating the pedestrians who walk over them. Notable exceptions were at the second pedestrian crossing where the shadow of a tree made it impossible to detect a couple of lines as shown in *Fig. 9*. However, locating the pedestrians was still good enough, as we can see in *Fig. 10*.

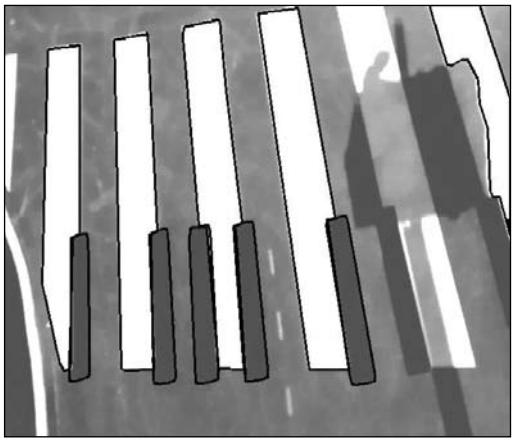

Figure 9: Detected contours of white and black keys on a real pedestrian crossing.

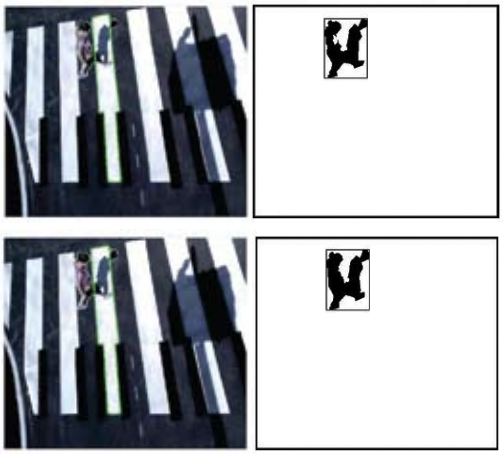

Figure 10: Comparing simple (up) and advanced (down) techniques for the separation of foreground and background showing in both cases the original image with the activated key next to the image with the located pedestrian.

In July 2010 a real installation of our application was set up during the 51st Jazz Festival in Ljubljana (*Jazz 51, 2010*). A pedestrian crossing with added black lines that resembled a real piano was set up near the entrance to the open air theatre Križanke where the Jazz Festival took place. The public could test the application and generate music by walking over the crossing (*Fig. 11*).

Figure 11. Art Installation Piano crossing during the 51st Jazz Festival in Ljubljana, July 2010. (Photo: Nada Žgank)

5. Conclusions

We created the art installation Piano crossing which applies the methods of computer vision to transform a normal pedestrian crossing into a music instrument. Computer vision serves as an interface between the physical world of people walking over a pedestrian crossing and the computer-generated musical tones. We developed an application that maps virtual piano keys to the stripes of the pedestrian crossing and that segments pedestrians from the background so that their mass centre can be computed and the corresponding musical tone generated. A particular challenge was that the installation is normally set up outside where the illumination changes frequently. We had to select and adapt appropriate computer vision methods for segmenting the stripes of the pedestrian crossing from the rest of the image and to segment the moving pedestrians from the background.

The application itself could be improved in several ways. The model of the piano keyboard is generated only at the beginning, if a camera is moved in any way due to the wind for example, the virtual piano keys become misaligned with the stripes of the pedestrian crossing. Musical tones are generated according to the alignment of the mass centre of a pedestrian with the virtual keyboard. A more advanced application could detect the actual position of user's feet on the virtual keyboard to make the correlation to piano playing even more convincing. The application could also implement the tracking of pedestrians instead of just locating them. In this way, a different musical instrument could be randomly assigned to individual pedestrians. We believe that this project is not interesting merely from an artistic viewpoint, but it could be also used for the promotion of events, services or products. On the other hand, it presents an interesting area for further research, namely how the transformation of motion into sound could help us in visual surveillance. Normal and routine events should sound the same, exceptional occurrences and unusual incidents should alert us with different sounds, rhythms or melodies.

References

ArtNetLab, 2010. http://black.fri.uni-lj.si, September 2010.

Batagelj, B. Ravnik, R. & Solina, F., 2008. Computer vision and digital signage. In Tenth International Conference on Multimodal Interfaces, pages 1-4. ACM, 2008.

BIG PIANO, 2010. http://en.wikipedia.org/wiki/Big_piano, December 2010.

DESIGN CROSSWALKS, 2010. http://www.crookedbrains.net/2010/01/design_09.html, December 2010.

DOUGLAS, D. H. & PEUCKER, T. K., 1973. Algorithms for the reduction of the number of points required to represent a digitized line or its caricature. Cartographica: The International Journal for Geographic Information andGeovisualization, 10(2):112-122, 1973.

DOVGAN, E., ČIGON, A., ŠINKOVEC, M., & KLOPČIČ, U., 2008. A system for interactive virtual dance performance. In 50th International Symposium ELMAR, volume 2, pages 475-478, Zadar, Croatia, 2008.

ESSA, I. A., 1999. Computers seeing people. AI Magazine, 20(2):69-82, 1999.

JAZZ 51, 2010. http://en.ljubljanajazz.si/photogallery/piano-crossing/, September 2010.

KIM, K., CHALIDABHONGSB, T. H., HARWOOD, D., & DAVIS, L., 2005. Real-time foreground-background segmentation using codebook model. Real-Time Imaging, 11(3):172-185, 2005.

KORTUM, P., editor, 2008. HCI Beyond the GUI : Design for Haptic, Speech, Olfactory, and Other Nontraditional Interfaces. Morgan Kaufmann, 2008.

LEVIN, T. Y., FROHNE, U. & WEIBEL, P., 2002. CTRL [SPACE], Rhetorics of Surveillance from Bentham to Big Brother. MIT Press, 2002.

LIPANJE, M., 2010. Klavir za pešce - ustvarjanje glasbe z računalniškim vidom /Piano crossing - generating music using computer vision. Master's thesis, University of Ljubljana, Faculty of Computer and Information Science, 2010.

PEER, P. & BATAGELJ, B., 2009. Art - a perfect testbed for computer vision related research. In M. Grgić, K. Delać, and M. Ghanbari, editors, Recent Advances in Multimedia Signal Processing and Communications, volume 231 of Studies in Computational Intelligence, pages 611-629. Springer, 2009.

PENTLAND, A. P., 1996. Smart rooms. Scientific American, 274(4):68-76, 1996.

PIANO STAIRS, 2009. http://thefuntheory.com/, 2009.

ROSENFELD, A. & PFALTZ, P., 1966. Sequential operations in digital picture processing. Journal of the Association for Computing Machinery, 12:471-494, 1966.

SARACENI, R., 2010.. http://www.walkingpiano.com, December 2010.

SOLINA, F., 2000. Internet based art installations. Informatica, 24(4):459-466, 2000.

SOLINA, F., 2005. 15 sekund slave in virtualno smu_canje / 15 Seconds of Fame and Virtual Skiing, 2005. Exhibition Catalogue. ArtNetLab, Ljubljana, 2005.

SOLINA, F., 2004A. 15 seconds of fame. Leonardo, 37(2):105-110, 2004.

SOLINA, F., 2004B., Artnetlab - the essential connection between art and science. In M. Gržinić, editor, The future of computer arts & the history of The International Festival of Computer Arts, Maribor 1995-2004, pages 148-153. Maska, Ljubljana, 2004.

SOLINA, F., & BATAGELJ, B., 2007. Dynamic anamorphosis. In Enactive/07 enaction in arts: Proceedings of the 4th International Conference on Enactive Interfaces, pages 267-270, Grenoble, France, 2007. Association ACROE.

SOLINA, F., BATAGELJ, B., & GLAMOCANIN, S., 2008. Virtual skiing as an art installation. In 50th International Symposium ELMAR, volume 2, pages 507-510, Zadar, Croatia, 2008.

STAUFFER, C. & GRIMSON, W. E. L., 1999. Adaptive background mixture models for real-time tracking. In IEEE Conference on Computer Vision and Pattern Recognition, volume 2, pages 246-252, 1999.

WALK THE TUNE, 2010. http://www.magdalena.org/en/gallery/entries/40199/details.html, December 2010.

WREN, C. R., AZARBAYEJANI, A., DARRELL, T. & PENTLAND, A. P., 1997. Pfinder: Real-time tracking of the human body. IEEE Transactions on Pattern Analysis and Machine Intelligence, 19(7):780-785, 1997.