IACV Project Report
Visual motion analysis of the player's finger

mailto:foobar@gmail.com Emanuele Paesano, - alessio.hu@mail.polimi.it
Alessio Hu, 10683050 - mailto:alessio.hu@mail.polimi.italessio.hu@mail.polimi.it
Supervisor: Vincenzo Caglioti - mailto:vincenzo.caglioti@polimi.itvincenzo.caglioti@polimi.it)

***Abstract*—We present a method for detecting the keys that are being pressed in a video where a keyboard instrument is played. We only consider the visual information, ignoring the audio, and use video data deploying a minimal camera setup, coming from a set of YouTube videos. Our method mainly exploits the change in the shape and lighting of the keyboard between frames, and uses techniques that include deep learning, image rectification and edge detection for processing the images.**

I. Introduction

A. *Problem Formulation*

The main goal of the project is to extract the sequence of notes of a song, using image analysis and computer vision tools and concepts. This must be accomplished starting from a video in which we can see a keyboard instrument being played. In this report we will mostly consider the piano as a possible keyboard instrument.

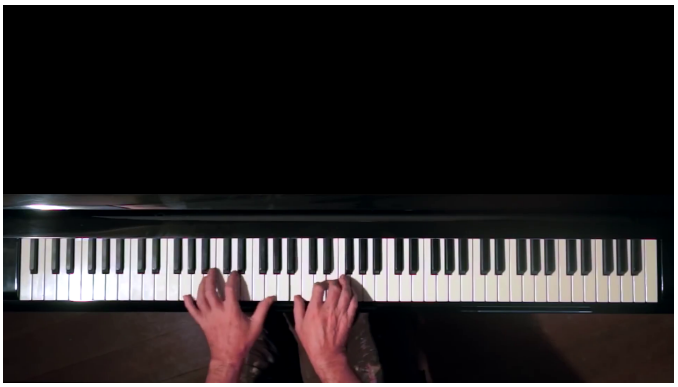


Fig. 1. Sample of one frame from a video of a piano player

In order to reconstruct the song, we will need to analyze the movement of the player's fingers and the changes that it causes in the image. Thus we will use image analysis tools to perform segmentation of player's hands and of the keyboard in every frame. This will allow us to reconstruct how their shapes change when a note is being played.

B. *State of the Art and previous works*

Automatic music transcription (AMT) is denoted as the process of automatically converting played music to a symbolic notation, such as a Musical Instrument Digital Interface (MIDI) file. In our project, unlike most of the work in this field, we are not going to process the audio of the video, but we will focus only on computer vision and image analysis techniques.

From the various papers which deal with similar tasks, [5] is the one which tries to achieve the same goal. In fact, [5] makes use of a minimal setup, made of a single camera, and exploiting only the visual content of the frames to detect the played keys. Also [4] succeeds in translating the video content to music, with remarkable results. However, [4] utilizes a camera configuration which is tailored for the purpose. More publications on similar topics include [1], which uses a depth sensor to detect the differences in depth for keys. [3] uses a combination of audio and video features, combined with deep learning, to achieve a reconstruction of the song. It also aims at assessing the accuracy of the pianist, for supporting piano students in their learning process. As opposed to [3], our project does not have a pedagogic purpose, but instead its only goal is to reconstruct the audio information starting from images. Similarly to [5], we will take our video data from a set of YouTube videos, thus considering a minimal camera setup.

II. Methodology

A. *Hand and Keyboard segmentation*

The first task of the project is to extract a boolean image mask, selecting only the hands of the player from the frames of a YouTube video, such as the one shown in Fig.1. We will do this on piano videos which are all taken from a static, overhead camera.

Then we want to do the same for the keyboard mask, which will allow us to segment the individual keys at a later stage. We will combine the keyboard and hand positions as a starting point to detect the pressed keys. We propose two alternative methods to achieve this.

*1) **Segmentation through manual HSV thresholding**:* HSV is an alternative representation of the RGB color model. Instead of being aligned to the way human vision perceives color-making (as RGB does), HSV models how colors appear under light. It is a cylindrical geometry, where hue is the angular dimension (starting from primary red at 0°) saturation is the distance from the center, and value is the height.

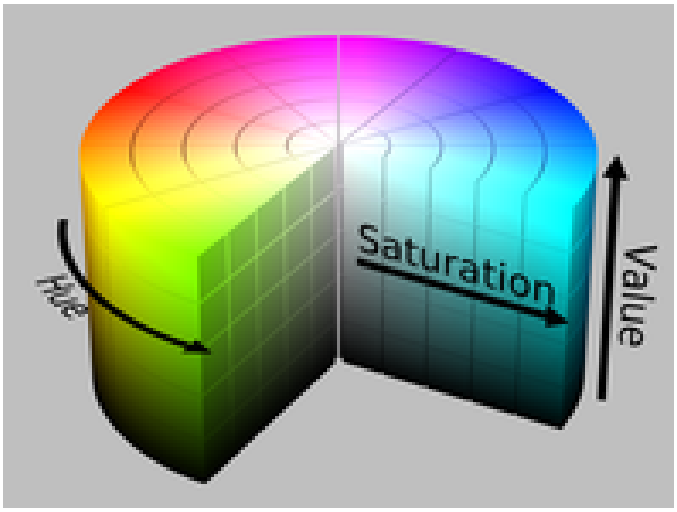


Fig. 3. HSV color scheme

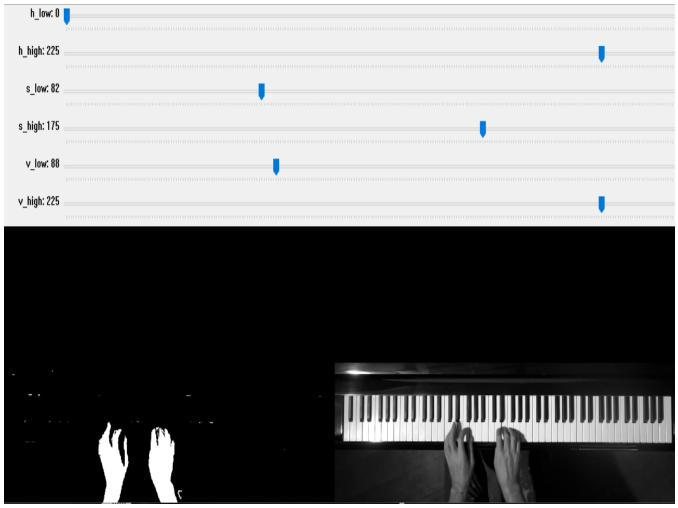


Fig. 4. Manual selection of HSV thresholds

This method leverages the difference in color (expressed in HSV) between the hands and the keyboard, in order to isolate each of them.
The simplest way to do this is to take one frame and manually select the HSV lower and upper thresholds that correspond to the color of the hands, and then do the same for the keyboard. The mask extracted this way is mostly valid for all of the frames, as the colors of the features should not change too much during the video. Fig.4 depicts the process of manually selecting HSV thresholds.

*2) **Segmentation through neural network**:* With this second method, we want to automatize the process by training a neural network to select the masks. This way we do not need to manually intervene every time, but we can instead extract the keys of the song by just feeding a video to our program.
Identifying the hand and keyboard masks is equivalent to assigning to each pixel of the input image one label among ”hand”, ”keyboard”, ”none”, so it is an image segmentation task.

- We first prepared the dataset by manually extracting from a single video frame both the original image and the corresponding hand and keyboard masks using the same method described in the first point.
- We then built a simple neural network with an encoder-decoder architecture following a U-Net like structure, a symmetric network with a contracting and expansive paths (Fig. I-B)
  - The encoder/contracting path is made of blocks with convolutional and max pooling layers
  - The decoder/expansive path is made of blocks with upsampling and concatenate layers, the latter of which implement skip connections and ease the task of identity mapping
- Finally, we trained the network by monitoring the accuracy and the mean intersection over union.

A reliable model of this kind has the advantage of being fully automatic, but obtaining the mask with this second method is slower if compared to the first one

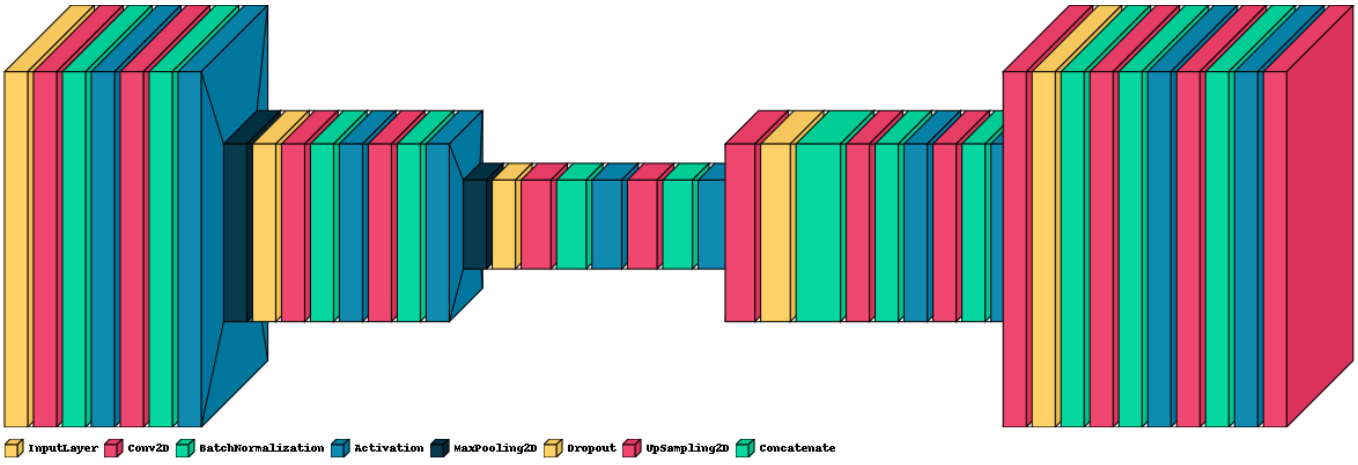


Fig. 2. U-Net architecture

B. *Keyboard Rectification*

Our aim is to identify the edges of the keyboard and of the individual keys automatically, by means of an edge detection algorithm. Before doing this, we want that all of the images to have a similar configuration. The ideal setting is to have a perfectly horizontal keyboard which we are looking at from above. We will achieve this result through image rectification. We will use the "opencv" python library for some computer vision operations. The pipeline we follow to rectify the images is the following:

1) Start from an image where the keyboard has been segmented, as in the previous section
2) To determine the shape of the keyboard, perform these steps:
   - Apply an edge detection algorithm to the image to find edges of the keyboard.
   - Dilate the image to connect the edges and obtain a continuous edge.
   - Use $cv2.findContours$ to find the contours, and sort them by the value of the area inside of each. The largest value should correspond to the area of the whole keyboard.
   - Approximate the largest contour to a 4-point polygon with $cv2.convexHull$, and draw only this one on a black frame.
3) Now we can compute the homography H to map this shape to a rectangle, taking as reference the top-left point of the polygon. This is achieved with $cv2.getPerspectiveTransform$.

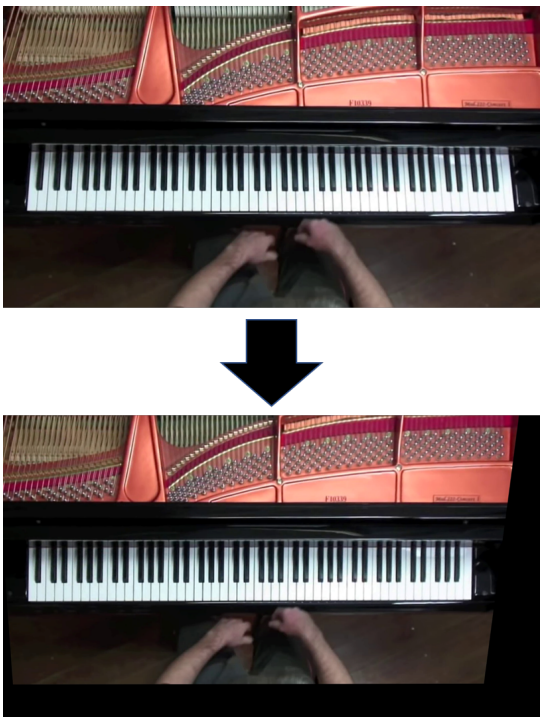


Fig. 5. Keyboard rectification

C. *Key regions retrieval*

We now go on to identify the regions of the individual keys. This will allow us to map every region to a single piano key that might be pressed in a given frame. We start from the rectified keyboard image, which we obtain by applying H to the segmented keyboard image.

Fig. 6. Rectified keyboard mask

We report the main steps of the process:

1) if the keyboard is in a vertical position (this happens often with instruments different from piano), we rotate it by 90°
2) rectify image by applying H found in previous section, and compute the keyboard mask using thresholds found in section A. This is a boolean image made up of white and black pixels.
3) Starting from this image, we identify the position and length of the black keys as that of all the continuous intervals of black pixels between two white pixels. We exclude shorter intervals that represent spaces between keys.
4) To find the white keys, we employ an heuristic where the black vertical space representing the start of a white key always starts at the middle point of a black key. Exploiting this fact, we find all of the x-coordinates where a white key starts.
5) When the space between middle points is very large, it means there are white keys that do not have a black key in between, such as the E-F and B-C keys.
6) We reconstruct the position of the rectangles corresponding to the white keys using the x-coordinates just found and the y-coordinates of the top and bottom of the keyboard.

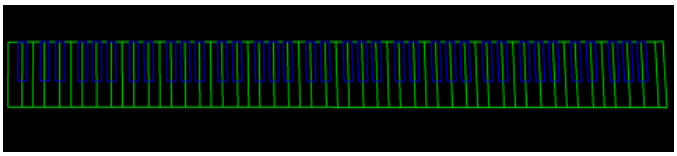


Fig. 7. Identified key regions

At the end of this process, we have divided the keyboard into regions which all correspond to either a black or white key. This is crucial to detect which key is being pressed at a given time of the song.

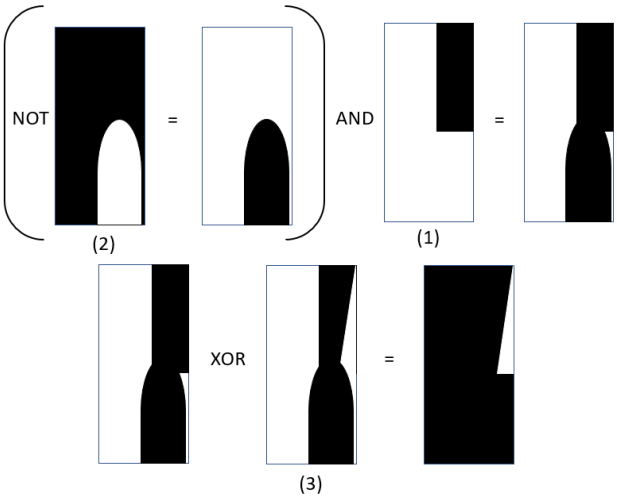
D. *Detection of Pressed Keys*

Once we have detected the regions of the keys, we can go on to detect which keys are being pressed at a given time. We will

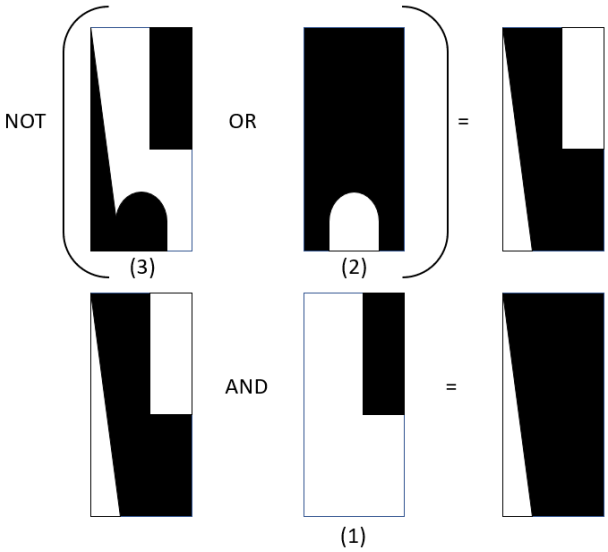
use the ”MediaPipe” python library to help us in identifying landmarks corresponding to finger joints in the image.

*1) **Selection of candidate keys through shadows**:* The first step consists in selecting the keys that might be pressed at a given time of the video. To detect them, we exploit the information that is contained in the change of shadows in the image. Namely, we apply bitwise operations to a considered video frame in order to obtain a new mask that isolates the regions of the newly formed shadows in each frame.
In particular, to isolate the casted shadows formed by pressing either the white or the black keys we use the static keyboard mask without the hands of the player (1), the hand mask (2) and the keyboard mask (3) of the current frame.

- Shadows of black keys:



- Shadows of white keys:



*2) **Exclude small shadows keys through Sobel and Hough lines**:* Additionally, we need to discard some of the detected candidates when the shadow is too small. In this case the shadow is probably caused by a more subtle movement that cannot be associated to the pressure of a key.
Since at this point we are working with a rectified binary frame, with the white regions corresponding to the shadows and rotated with the black keys on the top, we first apply a Sobel filter on the y-direction and then we use Hough transform to detect vertical lines that received enough votes. In this way we can discard cases in which shadows are too small: lines that get enough votes correspond to large enough shadow regions.

Fig. 8. Candidate keys through Sobel and Hough

The figure 8 shows the intermediate steps of this process on a test frame and and final results showing the candidate keys. This steps has to be repeated twice, first to obtain candidate white keys then for candidate black keys. We notice that not only we require different thresholds for the two cases, but also for different videos the thresholds have to be adjusted as well, due to different lighting conditions and possible less accurate segmented masks from unfocused and low quality videos.

*3) **MediaPipe to detect fingers**:* We employed Google’s MediaPipe library to support us in our task. MediaPipe offers a set of libraries and tools for quickly applying artificial intelligence and machine learning techniques in applications. We are interested in the part of MediaPipe related to computer vision tasks. In particular, its ”Hand Landmarker” tool can identify any hand contained in the image, with a number of landmarks corresponding to specific points of the hands, allowing to track its position and movement. Out of all the landmarks offered by the library, reported in Fig 9, we will consider those associated to the fingertips.

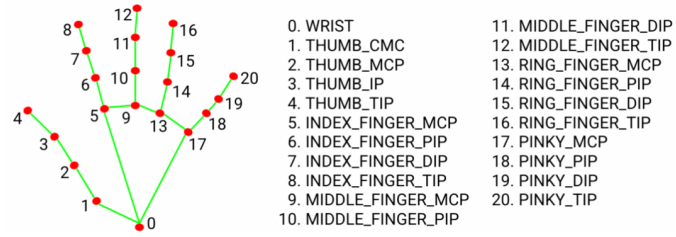


Fig. 9. Mediapipe Hand Landmarker

In particular, we consider the following landmarks:

`"THUMB_TIP","INDEX_FINGER_TIP", "MIDDLE_FINGER_TIP","RING_FINGER_TIP", "PINKY_TIP","PINKY_DIP", "THUMB_IP".`

Fig. 10. Detected Hand Landmarker

*4) **Final selection of pressed keys**:* Once we know the position of the fingers, we can compare it to the position of the candidate keys (the ones featuring shadows, filtered with the method using Hough lines). A key is identified as pressed if the positions are compatible, meaning that:

- The key is a candidate keys, identified previously through shadows, Sobel and Hough lines
- The region key contains the point which identifies the finger position, in particular to this aim, we use the function *cv2.pointPolygonTest* from the opencv library, to which we feed the landmark point and the polygon corresponding to the candidate key. The function returns True if the point is inside the polygon, its distance otherwise.

By applying this to every frame of the video, we are able to extract the sequence of notes of the song. We finally store the notes in a json file and convert it to MiDi as a playable format for the song.

III. Results

We applied our methodology on a video containing a top view of a full piano keyboard [2]. While the rectification, retrieval of key regions and pressed keys detection remain the same, the masks segmentation part can be carried out with two different methods as we have presented:

*1) **Using the manual method**:* Imprecise masks may be generated if the video contains other objects whose colors resemble either the white keys' colors or the player's skins' colors.

*2) **Using the neural network for segmentation**:* With a large dataset and more computing power we could have a more accurate model for segmentation, but for demonstration purposes we used just use a small sample by extracting some frames from a dozen of different videos depicting keyboards. For each frame we stored the corresponding hand and keyboard masks in a separate file.

As we can see in Fig. 11, the network reaches a high accuracy and meanIoU within 20 epochs. But if we keep training the network, due to the few data we have, it will worsen its overall performance, hence we used early stopping and dropout layers to avoid overfitting.

Fig. 12 shows a the performance of the network on a sample image.

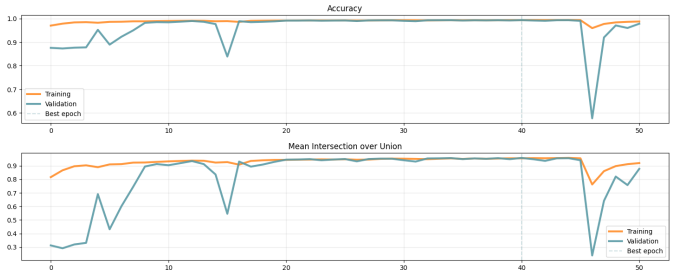


Fig. 11. Accuracy and mean intersection over union over the epochs

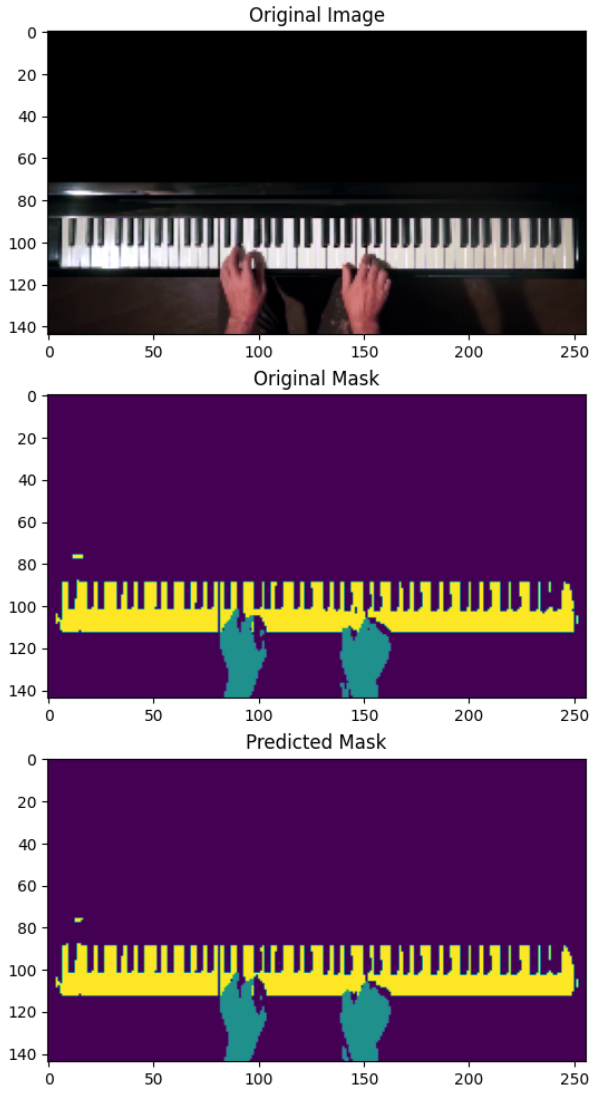


Fig. 12. Segmentation test on a frame

We [2]

IV. Conclusions

The algorithm constitutes a promising starting point, providing a reconstruction which closely resembles the sound of the original song. While our ears can immediately recognize the song, the reproduction is far from perfect. Some factors

that contribute to lowering the accuracy of the model might be:

1) The manual mask selection may not be accurate.
   In particular, when we have objects in the image whose color is very similar to those of the hands or keyboard, it is impossible to segment them by selecting HSV thresholds manually. The neural network segmentation can possibly overcome this issue by providing a more precise thresholding, but it is subject to other problems.
2) The training data for the neural network is not sufficient.
   Our model for detecting the keyboard mask tends to overfit the training set, meaning its accuracy could likely be improved with more data. A correct segmentation of the keyboard is crucial for improving the performance of the algorithm.
3) The pipeline for detecting key pressure is sensitive to changes in the shadowing in the image.
   The main factor here is the movement of the pianist's hands, which can cast shadows on top of the keys. In a lower light setting, these shadows may not be distinguished from the ones produced by a pressed key. Thus, the model can be prone to mispredict, especially in under low light conditions.

Additionally, it is possible to extend the algorithm to accommodate for a wider range of keyboard instruments. In our tests we selected the accordion as an alternative instrument. The results showed that our method is able to segment the hands and keyboard correctly, but struggles when it comes to rectifying the image and selecting the pressed keys. The main issue here is the continuous motion of the instrument from frame to frame, as opposed to the static keyboard of the piano. In Fig.13 we provide some of the results obtained by testing the algorithm on an accordion video. We will leave further development in this area as possible future work.

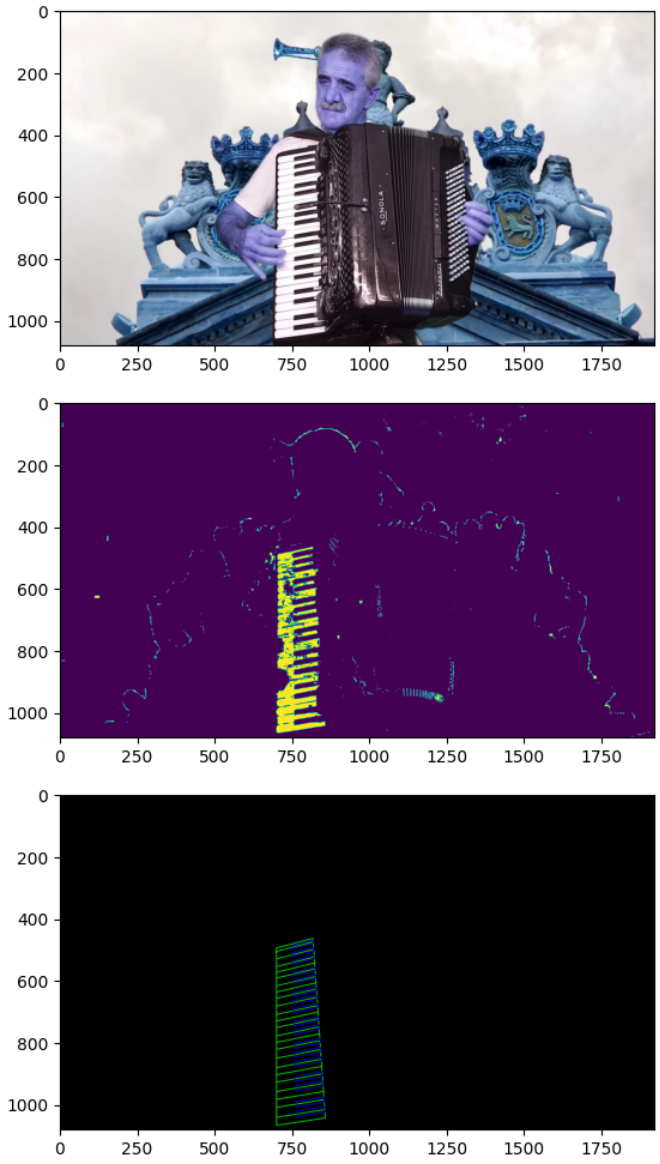


Fig. 13. Keyboard segmentation on accordion

REFERENCES

[1] M. Hashimoto A. Oka. Marker-less piano fingering recognition using sequential depth images. *The 19th Korea-Japan Joint Workshop on Frontiers of Computer Vision*.

[2] https://youtu.be/rxZOrefeGLE.

[3] Yupeng Gu David Cartledge David J. Crandall Jangwon Lee, Bardia Doosti. Observing pianist accuracy and form with computer vision. *School of Informatics, Computing, and Engineering, Indiana University Bloomington*.

[4] Howard Cheng Mohammad Akbari. Real-time piano music transcription based on computer vision. *IEEE Transactions on Multimedia*.

[5] Potcharapol Suteparuk. Detection of piano keys pressed in video. *Stanford University*.