Politecnico di Milano
Computer Science and Engineering
099993 - Image Analysis And Computer Vision

Small Project F12

Visual reconstruction of played music

2st semester 2024/2025 - Professor: V. Caglioti

Ignacio Bascuñán C. - 10984708

Contents

Abstract

Computer vision techniques can infer which piano keys are being pressed from a silent video of a performance. We present an end-to-end pipeline that rectifies the keyboard, maps key polygons, detects fingertips with MediaPipe, and validates presses via background-subtracted photometric differences. The system recovers key presses and exports the transcription as a MIDI file.

1 Introduction

For the final project in our Image Analysis and Computer Vision course, we were assigned a very specific challenge: given a silent video of someone playing the piano, determine purely from the images which keys are being pressed and when. At first glance it sounds straightforward, but once you dive in you realize how many visual quirks work against you: keys half hidden by moving hands, glossy reflections that shift with every light change, and camera angles that stretch the keyboard into odd trapezoids.

In this report, we explain the classical (non-deep-learning) pipeline we built in Python and OpenCV to tackle the task. We point out what went right, what failed spectacularly (black keys, I'm looking at you), and where the method still needs improvement before it could be used in a real musical setting.

2 Related Work

Early attempts to localize piano-key activations relied on template matching and frame differencing techniques applied to static key patches [1]. With the development of deep learning, convolutional architectures such as **DeepKey** [2] and **transformers** [3] learned a direct mapping of RGB frames and spectrogram sounds to note onsets respectively, but typically required a synchronized audio track to provide reliable supervision and alignment. Subsequent hybrid approaches exploited both modalities: for instance, **Onsets and Frames** [4] achieve state-of-the-art transcription accuracy by fusing visual cues with microphone input, but cannot operate when audio is absent.

Complementary progress in hand-pose estimation, exemplified by MediaPipe Hands [5], now enables millimeter-level fingertip localization in real time, offering strong spatial priors for piano interaction. Nevertheless, the explicit integration of such pose information into classical key-classification pipelines is still limited in the literature.

The present work addresses this gap by proposing a fully vision-based, lightweight approach that combines geometric keyboard rectification with fingertip-guided photometric analysis, without relying on deep models or paired audio data.

3 Methodology

The proposed pipeline is decomposed into an *offline calibration* phase operating on a still reference image (Figure 1) and an *online* phase operating on each RGB frame (Figure 2).

3.1 Keyboard Calibration

Geometric rectification. Given a manually captured reference frame I_r without hands, we extract dominant horizontal and vertical edges using Canny filtering followed by probabilistic Hough transforms. The angle degree for horizontal and vertical lines allows for a slight variation to account for image distortions and rotations. The 4 most external lines are chosen as the keyboard boundary. Their intersections define a quadrilateral whose homography $H \in \mathbb{R}^{3\times3}$ maps the piano surface onto a fronto-parallel rectangle, undoing perspective distortion.

Key segmentation. In the rectified domain I'_r, horizontal projection of a Gaussian-blurred luminance channel reveals periodic valleys corresponding to gaps between keys. Thresholding these projections yields vertical stripes; their centroids delimit white keys. A user-supplied index k_0 identifies the left-most key, fixing the chromatic pattern white, black.

Inverse mapping with H^{-1} transfers each key polygon K_i back to image space for subsequent frame analysis (Figure 3).

3.2 Frame-wise Key-Press Detection

For each incoming RGB frame F_t (30 or 60 fps), we perform:

1. **Hand localization**: MediaPipe Hands returns up to 21 landmarks per detected hand. Fingertip indices yield image projections S_t.
2. **Candidate mapping**: For each fingertip $s \in S_t$ we detect all the keys that the fingertip may be touching.
3. **Photometric validation**: Skin pixels are masked via joint YCrCb and HSV thresholding, and removed from both the reference image (image without hands)

and the current RGB frame. The absolute difference of the two is then compared and thresholded (to account only for relevant changes), and with this we calculate the count of changed pixels. If the percentage of changed pixels is more than a parametrized threshold (10% - 15%) then the key is said to be pressed. This method exploits the fact that a pressed key exhibits darker shading and altered specular highlights relative to its pristine state.

3.3 MIDI Synthesis

Detected (t, i) tuples are clustered by temporal proximity ($<$50 ms) to yield note on/off events, which are finally written to a `*.mid` file using the `mido` library. The tempo of the song is deduced from the fps.

Algorithm 1 Key Extraction and Press Detection

Require: Reference image I_r, video frames F_t, first-key index k_0
Ensure: Press events $P = (t, k)$, MIDI file

Calibration: Detect borders $\rightarrow$ compute H; segment keys $\mathcal{K}$
for each frame F_t **do**
    $S_t \leftarrow$ MediaPipe fingertips
    for each $s \in S_t$ **do**
        $i \leftarrow \text{key\_index}(s, \mathcal{K})$
        if $i \neq \emptyset$ and absolute difference $> \tau$ **then**
            $P \leftarrow P \cup (t, i)$
        end if
    end for
end for

4 Implementation Details

The prototype is implemented in Python3.10 and OpenCV4.11 on an Inteli7-10700K CPU and an NVIDIA RTX-3060Ti GPU. MediaPipe0.10 delivers real-time hand tracking ($<$,8 ms/frame). All parameters (e.g. Canny thresholds, absolute difference threshold, etc.) are exposed in a JSON configuration file for reproducibility.

5 Experimental Evaluation

5.1 Dataset

Seven HD clips from YouTube were selected and automatically trimmed to segments of 60 s to 120 s. For each video, a manual reference screenshot was taken to find a moment in which no hands were present on the keyboard.

5.2 Output

The output obtained for each video were:

- A variety of debugging images of the transformation process
- A JSON file with the timestamps of each note being pressed
- A MIDI file with the simulated output song, which can be heard on Pianotify

5.3 Metrics

Due to poor output quality and results, the metrics used for validating and testing were human feedback. 3 people were asked to hear both the original audio and the simulated audio, and they were asked to classify them into 5 categories:

1. Impossible to recognize
2. Some notes and rhythms seem to match the original song
3. The song is recognizable
4. The song not only is recognizable, but is pleasant to hear
5. Just some mistakes were made between the original song and the simulated output.

5.4 Results

Table 1 summarizes the obtained results. Only two songs were recognizable, one was impossible to recognize, and the rest were in between unrecognizable and barely recognizable.

Video	Person 1	Person 2	Person 3	Mean
Video 1	1	2	2	1.67
Video 2	2	2	2	2
Video 3	1	1	1	1
Video 4	4	3	3	3.33
Video 5	1	3	3	2.33
Video 6	3	3	4	3.33
Video 7	2	1	2	1.67

Table 1: Human Feedback Results

6 Discussion

Qualitatively, the system was able to recover some melodic contours and rhythmic patterns for some pieces. The interesting thing is that the quality of the output is not directly dependent on the tempo and complexity of the piece, but rather on the illumination and angle of the video. Furthermore, failures are more predominant around extremely rapid arpeggios and passages where both hands occlude the keys.

Key limitations include:

- Sensitivity to specular reflections and shadows, mitigable via adaptive thresholding or polarizing filters.
- Inability to resolve overlapping fingertip projections (chords) when occlusion persists for multiple frames.
- Neglect of temporal coherence; a Kalman filter or hidden Markov model could suppress flicker.

7 Conclusion and Future Work

This study demonstrates that a classical pipeline, based on hand-pose priors and image filtering, can partially recover played notes from silent piano videos. While white-key detection is moderately reliable, black-key accuracy remains low. Future work could explore (i) temporal convolutional networks trained on synthetic renderings, (ii) multi-cue fusion with optical-flow-based key motion, and (iii) self-supervised fine-tuning to individual performers using readily available audio.

This study not only shows the potential that a traditional (non-deep learning) method can have on detecting key presses, but also its limitations.

- This method struggles on adapting to each different case and scenario, requiring to do a manual parameter calibration for each.
- Black keys are harder to detect due to lack of shadows; they require a greater level of calibration for accurately catching them.
- Hand mask is not completely reliable as depending on the video lighting and colors they can get mixed with the same keys. This can be improved by building a dynamic mask accounting for the position of the detected hands' skeleton.

Many things could be optimized in this method, specifically working on the area of adaptive parameter calibration for improving accuracy. Furthermore, even if the method is relatively fast ($\sim$ 500ms per frame) it can be optimized to run faster by merging filters and methods, and by avoiding redoing unnecessary computations.

A Supplementary Figures

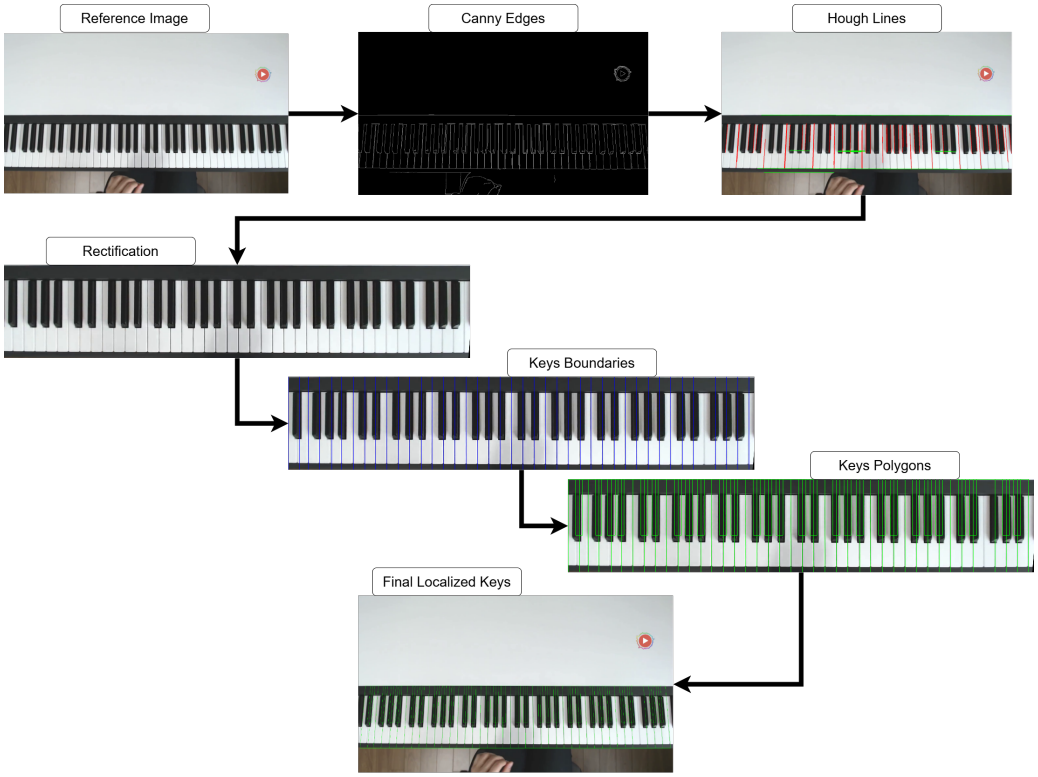

Figure 1: Overview of the proposed offline pipeline.

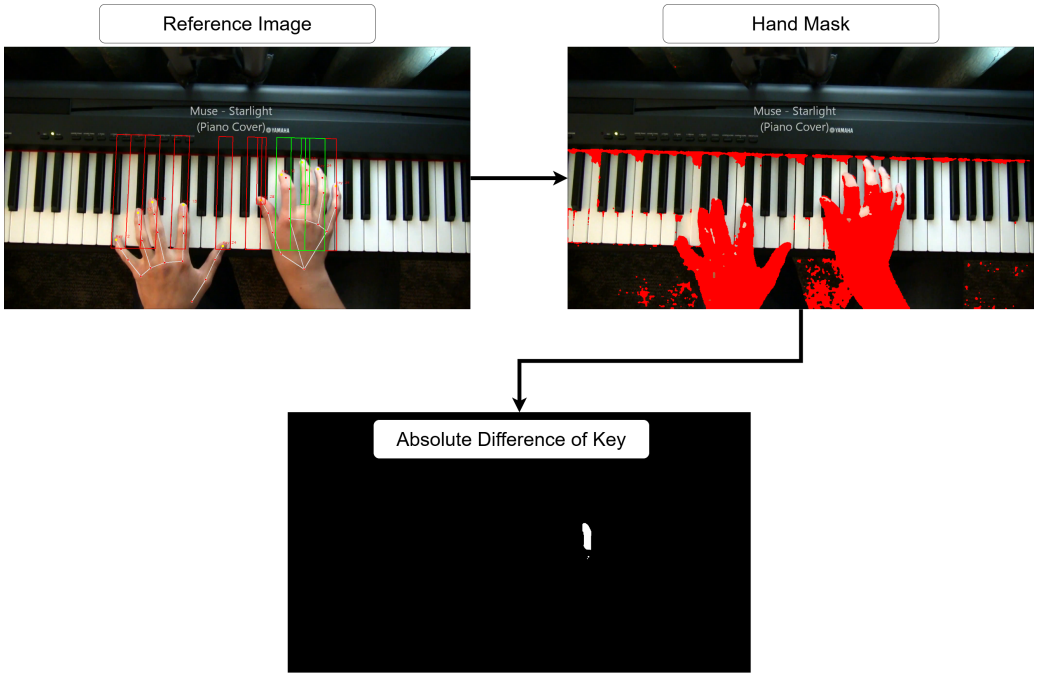

Figure 2: Overview of the proposed online pipeline on key 34.

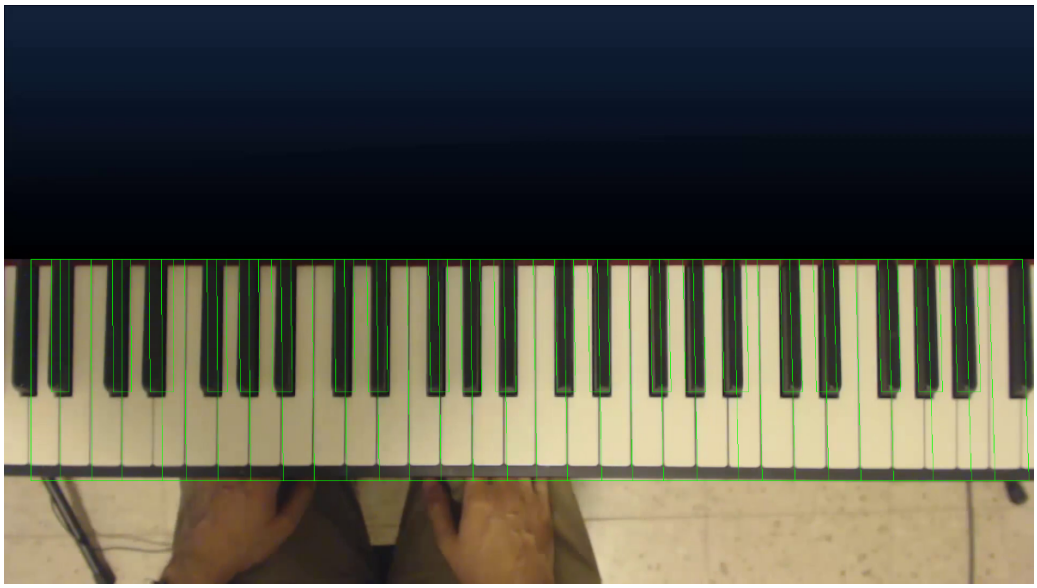

Figure 3: Final key mapping on original image.

References

[1] D. Blostein and L. Haken, "Template matching for rhythmic analysis of music keyboard input," [1990] Proceedings. 10th International Conference on Pattern Recognition, Atlantic City, NJ, USA, 1990, pp. 767-770 vol.1, doi: 10.1109/ICPR.1990.118213.

[2] Smith, Rory & Burghardt, Tilo. (2018). DeepKey: Towards End-to-End Physical Key Replication From a Single Photograph. 10.48550/arXiv.1811.01405.

[3] Hawthorne, C., Simon, I., Swavely, R., Manilow, E., & Engel, J. (2021). Sequence-to-Sequence Piano Transcription with Transformers. arXiv (Cornell University). https://doi.org/10.48550/arxiv.2107.09142

[4] Hawthorne, Curtis & Elsen, Erich & Song, Jialin & Roberts, Adam & Simon, Ian & Raffel, Colin & Engel, Jesse & Oore, Sageev & Eck, Douglas. (2017). Onsets and Frames: Dual-Objective Piano Transcription. 10.48550/arXiv.1710.11153.

[5] Zhang, F., Bazarevsky, V., Vakunov, A., Tkachenka, A., Sung, G., Chang, C., & Grundmann, M. (2020). MediaPipe Hands: On-device Real-time Hand Tracking. ArXiv, abs/2006.10214.