# Virtual Musical Instruments

Abhinav Rastogi, Ameya Joshi

***Abstract*—The virtual musical instrument is an interface that allows users to simulate a musical instrument by printing a template on a sheet of paper, placing it in view of the webcam of their laptop, and running the console application. The user then 'plays' the virtual instrument as if it were a real one, and appropriate music is generated. This is achieved by calibration of the template, detecting the marker position, identifying the hit position and playing notes corresponding to that position. We demonstrate this process for three different 'virtual instruments' and also report the accuracy of estimating the hit position.**

***Index Terms*—Object detection, calibration, homography, human computer interaction, tangible interaction**

## I. Introduction

Computers have become pervasive today. However, even now, we interact with them using traditional input methods like mouse, keyboard and more recently the touch screen. The digital image processing class introduced us to techniques that could be used to quantify human gestures and movements. We realized that using these techniques, we could build an intuitive, low-cost and tangible interface.

Everybody likes to play music, if not, to at least listen to music. One of the barriers to playing music for recreation is the high cost of musical instruments. Many people want to play a piano once in a while, but are not always ready to spend more than a hundred dollars for that sake. Also, musical instruments require space and regular maintenance. You may feel like trying a different musical instrument after playing the old one for a long while. In order to do that, you'll have to buy a new one. These plausible situations motivated us to apply the image processing skills we learned in class to the problems mentioned above. We came up with a virtual musical instrument, which is nothing but a template printed on a piece of paper. Our set up also doesn't require you to buy fancy hardware. All you need is a laptop that has a web camera.

The application is run on the laptop and the user plays the virtual instrument as if it were a real one. Consider the example of drums, where the user is hitting the paper template with a 'stick' (a marker). The camera captures the movement of the stick, the application then estimates the hit position and time, and the sound corresponding to that hit position in the real-world musical instrument is then played through the speakers.

## II. Related Work

There have been numerous projects that have explored the use of augmented reality, three-dimensional interfaces, physical and tangible interaction for generating music. One such project is the 'Augmented Groove' [1]. In the augmented groove, users can play music together, with or without traditional musical instruments, simply by picking and manipulating physical cards on the table. The physical motions of the cards are mapped to changes in musical elements such as timbre,pitch, rhythm, reverb and others. At the same time, users wearing a lightweight head-mounted display (HMD) can see 3D Virtual instruments attached to the cards whose shapes, color and dynamics reflect aspects of the music controlled by the visitors. The music in a sense becomes a physical and tangible object, something we can touch and see as part of our physical environment.

The reacTable [2] is a multi-user electro-acoustic musical instrument with a table-top tangible user interface. It is hence very similar to our setup, which also uses a table placed in view of the laptop webcam. In the reacTable, several simultaneous performers share complete control over the instrument by moving physical artefacts on the table surface while constructing different audio topologies in a kind of tangible modular synthesizer or graspable flow-controlled programming language. The instrument hardware is based on a translucent round table. A video camera situated beneath, continuously analyses the table surface, tracking the nature, position and orientation of the objects that are distributed on its surface. The tangible objects, which are physical representations of the components of a classic modular synthesizer, are passive, without any sensors or actuators; users interact by moving them, changing their position, their orientation or their faces. These actions directly control the topological structure and parameters of the sound synthesizer. A projector, also from underneath the table, draws dynamic animations on its surface, providing a visual feedback of the state, the activity and the main characteristics of the sounds produced by the audio synthesizer. Our design does not currently use any active illumination or projector that augments the template.

## III. Virtual Music Player

### A. Process flow

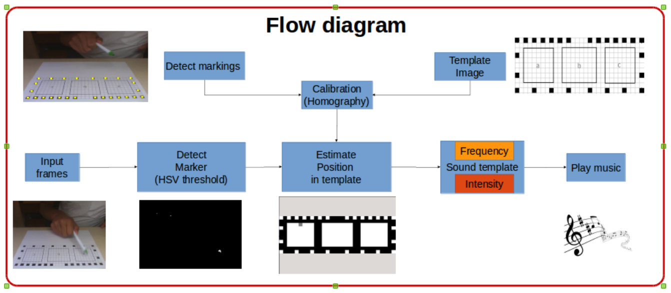


Fig. 1: Flow diagram of virtual music player

The application first calibrates the position of the markers. Each template has a set of dots located on the bordering grids, which are used for calibration. The process of calibration essentially involves calculating the homography from the image observed from the camera to the template image stored on the system. Once calibration is over, the application detects the marker in each input frame. This results in a time series of marker positions, with a few possible gaps due to limitation of the camera. This time series is used to obtain the instants when the marker hits the paper template. The homography calculated during calibration is then used to estimate the location of the marker in the grid by making use of the marker coordinates in the input image at these instants. The music corresponding to the detected position is then played on the speaker.

### B. Calibration

We use morphological image processing for locating the black dots present on the border. We do binary thresholding of the input image using Otsu's method. The result for this operation can be seen in Fig 2(b). Since the page occupies the largest area in the input image, the largest connected component is expected to belong to the the page. The detected region is then eroded with a $5 \times 5$ mask to remove spurious detections. Then we look for connected components located inside the detected region corresponding to the page. The different connected components have been shown in Fig. 2(c), where black color denotes the largest connected component and the remaining ones are shown in white.

Amongst the detected components, we do size based filtering to detect the ones corresponding to the marker. The size thresholds used are normalized with respect to the frame size of the input image so that the detection is independent of the camera used. Next we associate the detected key-points with the corresponding points in the template image saved in the system.

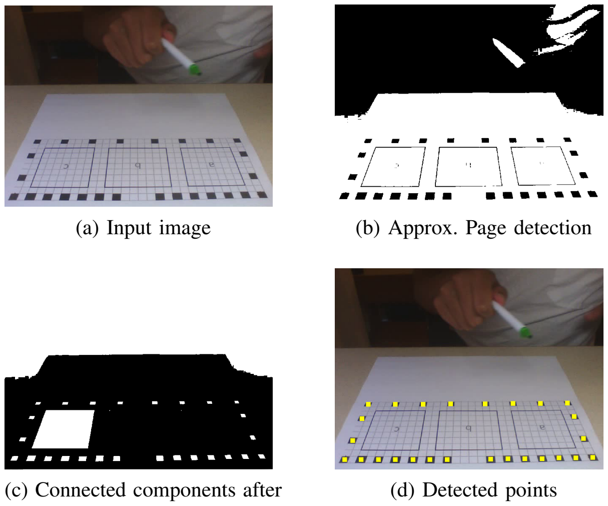

(a) Input image (b) Approx. Page detection
(c) Connected components after erosion (d) Detected points

Fig. 2: Detection of template key-points during calibration

For this we take advantage of the fact that all templates have the same border. We sort the detected points based on the y-coordinate in the input image in ascending order. The points are then clustered based on their y-coordinate. Then points within each cluster are sorted according to x-coordinate to get the appropriate correspondences. Due to structure of the template, the clustering operation is quite trivial since we know that there will be 4 clusters having 8, 2, 2 and 14 points respectively.

We tried to do SIFT based matching of the two frames and were able to get it working on MATLAB using VLFeat package but we were not successful in observing the same performance on our OpenCV based C++ implementation. Our detection fails when there are fewer or more than 16 detections in the image. To mitigate this, we loop through the initial images till we obtain the desired number of key-points at appropriate locations. During numerous tests conducted in various lighting conditions (yellow light, day light, indoor light) we observed that our calibration process works flawlessly if the background doesn't contain black dots.

### C. Marker detection, localization and tracking

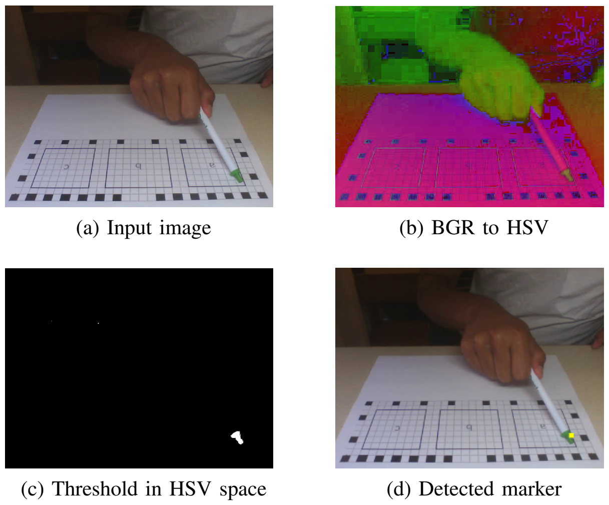

(a) Input image (b) BGR to HSV
(c) Threshold in HSV space (d) Detected marker

Fig. 3: Detection of marker from a frame in webcam feed

We use a standard pen with a green colored cap on its end as a marker. We observed that detection is easier in HSV space and hence we first convert the BGR space image to HSV space. Then we set a threshold in the HSV space corresponding to the following values LowHue = 39, HighHue = 80, LowSaturation = 57, HighSaturation = 157, LowValue = 0, HighValue = 255. These thresholds were calculated after tests under different conditions. This threshold allows us to selectively pick only the green parts of the webcam feed. However there is some background noise that lies within the same limits. To remove it, we find the connected components of the output and retain only the largest connected component that has a size of 150 pixels.

Once we detect the marker, we need to find the position of the tip. This is done by taking the mean along the x-axis of the image ($x_0$), of the pixels that constitute the largest connected component and by finding the maximum of the y-values of the pixels that make up the largest connected component ($y_m$). We say that $(x_0, y_m)$ is the position of the tip of the marker.

On testing, we found that change in orientation and lighting conditions, sometimes leads to no connected component of size 150 (normalized by image frame size). Also, if we decrease the size from 150, we may pick up noise instead of the marker. Due to the above reasons, there are certain frames in the feed, in which no marker is detected. This creates a necessity to predict or estimate the marker position when a detection is not available. Hence to get an estimate of the marker in every frame, we linearly interpolate the last two marker positions to arrive at the current one, in case we fail to detect the marker in the current frame. Using more detections from the past for linear interpolation might increase our chances of missing a maxima. Other interpolation techniques, perhaps based on past observations of user behavior are a good direction for future work.

### D. Hit detection

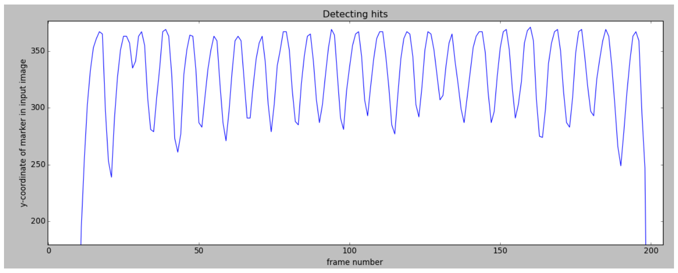


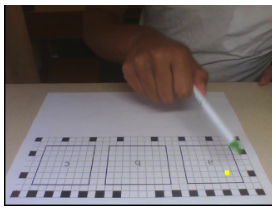
Fig. 4: Plot of $y_m$ versus frame number. The local maxima correspond to hits

Since the sound should be produced only when the marker tip hits the paper template, we should be able to locate the hit-moment. In our simplified approach, we associate the maxima in the observed y-coordinates of the marker as hit instants. Since detection of maxima requires knowledge of future samples, we introduce a delay of 2 frames.

We perform maxima detection using a sliding-window which has been implemented using a queue of size 5. As soon as the current marker tip position is received, we insert the y-coordinate of the marker tip $y\_current_m$ at the end of the queue and pop out the oldest element from the queue. We keep a history of 2 $y_m$ values which are ahead of the current value in the queue. When the center element of the queue is strictly greater than all elements to its left and is greater or equal to all elements to its right, a maxima is detected. The strict inequality is needed to prevent 'ringing' artifacts (repetition of the same sound) which happens when successive marker positions have the same value.

### E. Sound playing

Once we detect the hit position in the camera coordinates, we use the homography to find the corresponding location in the printed template. The sound template consists of two layers. One layer gives the frequency for every pixel in the instrument template and the second layer gives the intensity for every pixel in the instrument template. We can code the sound template in a way that resembles the spatial, frequency and intensity variation of a traditional instrument like drums. Hence once we get the hit-location, the frequency and intensity of a traditional musical instrument corresponding to that location can be played. The tones are stored as audio files and are played in a separate process by creating a pipe so that the detection program doesn't halt.


(a) Input image

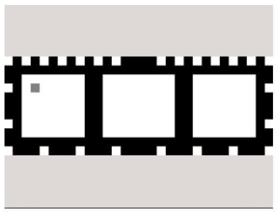

(b) BGR to HSV

Fig. 5: Detection of marker from a frame in webcam feed. The difference between the pen position and detected position is a result of frame lag introduced by maxima detection

### F. Results

We tested our implementation on 3 vidoes made for 3 different instrument templates. Further to test the limits or our application, we checked its performance when the speed of hitting is increased. Our results are tabulated below. It can be seen that at normal hit speeds, we detect all the hits within a distance of one grid. As the speed is increased, the performance deteriorates. The application is also fairly robust in terms of the quality of light available but not the intensity. Our test videos were made in indoor light whereas we demonstrated our system in outdoor light. Good performance was observed in both these cases.

| Test Video | Avg frames per hit | Total hits | Correct | D = 1 | D >1 | Accuracy |
|---|---|---|---|---|---|---|
| 1 | 11.67 | 36 | 31 | 5 | 0 | 86.1% |
| 2 | 11.67 | 18 | 9 | 9 | 0 | 50 % |
| 3 | 11.67 | 18 | 13 | 5 | 0 | 72.2 % |
| 4 | 10 | 36 | 29 | 7 | 0 | 80.56 % |
| 5 | 11.67 | 18 | 16 | 2 | 0 | 88.89 % |
| 6 | 11.67 | 18 | 17 | 1 | 0 | 94.44 % |
| 7 | 10 | 33 | 28 | 5 | 0 | 84.85 % |
| 8 | 9.13 | 22 | 14 | 8 | 0 | 63.63 % |
| 9 | 8.57 | 22 | 20 | 2 | 0 | 90.91 % |
| 10 | 6.67 | 36 | 18 | 12 | 6 | 50 % |
| 11 | 3.75 | 24 | 12 | 2 | 10 | 50 % |
| 12 | 5 | 24 | 7 | 3 | 14 | 29.17 % |

### G. Future work

Calibration can be made more robust against lighting and background changes. We can use SIFT feature matching followed by RANSAC to get the homography. This can also be used for automatic detection of templates from a fixed template database.

The latency of the entire process can be reduced. One way to do this can be storing larger number of historical values of the marker position and using them to predict in real-time (without looking at any of the future values) that the user is going to hit the paper.

Another improvement can be made in terms of the accuracy of the localization and timing of the hit. This can be achieved by detecting the shadow of the marker on the template and estimating the position and hit moment by determining when the marker and its shadow overlap or come close.

## IV. Conclusion

We have designed, implemented and tested a virtual musical instrument system. Our system is low-cost and the user needs to print just one sheet of paper for every instrument he/she wishes to play. The latency of the process is small, and the sound output is near real-time, though there is still room for improvement. Our accuracy for the hit localization is 80% for typical user behavior. We also successfully demonstrated our system at the poster session.

## Appendix A
## Contributions

- **Abhinav** - API design, Calibration, Hit detection, Audio playing
- **Ameya** - Proposal, Object detection and tracking, Testing and Results
- **Joint Efforts** - Poster, Report, Template Design

## Acknowledgments

We would like to thank Dr. Haricharan Lakshman for his valuable suggestions which guided the design and implementation of this project. We would also like to thank Prof. Bernd Girod, Prof. Gordon Wetzstein and the TAs for their guidance and support.

## Source code

https://github.com/abhirast/MusicVision

## References

[1] Poupyrev, I., Berry, R., Kurumisawa, J., Nakao, K., Billinghurst, M., Airola, C., & Baldwin, L(2000). Augmented groove: Collaborative jamming in augmented reality. In ACM SIGGRAPH 2000 Conference Abstracts and Applications (p. 77)

[2] Kaltenbranner, M., et al. ”The reactable*: A collaborative musical instrument.” Enabling Technologies: Infrastructure for Collaborative Enterprises, 2006. WETICE’06. 15th IEEE International Workshops on. IEEE, 2006.