AR Sampler using Image Processing

Prateeksha Singh
Surabhi Pawar
Smitesh Sawant

Table of Contents

LIST OF FIGURES

1. 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 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. All of the above is brought about by the use of Image Processing, the domain where the images captured from sources such as videos or web camera and are processed using image enhancement techniques to filter noise and to enhance the area

of interest. Image processing is nothing but to analyze and manipulate a digitized image, especially in order to improve quality of that image and use it for further operations.

1.1 PROBLEM DEFINITION

A primary goal of our project is to create a system which can identify specific human gestures and use them to convey information for controlling device and by implementing real time feature recognition a user can control a software synthesizer by tapping a marker on a printed sheet of the predefined template front of a video camera which is linked to the computer, or, even more convienient, a built in laptop webcam. A primary goal of this gesture recognition research is to create a system which can identify specific human gestures for controlling the third party applications.

Our project also covers various issues like what are features in an image that help in distinguishing them, their classification, their role in implementing a recognition system for controlling features like type of sound to be played, system architecture concepts for implementing an AR system, major issues involved in implementing realtime audio feedback, and future scope of local augmented reality. For implementation of this system real-time marker tracking, thresholding and extraction algorithm, and feature extraction are used.

1.2 RELEVANCE OF THE PROJECT

Augmented Reality Musical Instruments is the system which is used where the new technologies are emerging with the use of gesture control mechanisms to operate systems such as Games and operating Web Pages with the hand-based motion mechanism. The system uses the web camera to detect the hand gesture by calculating the threshold value. According to the movements tracked by the web camera the system allows the applications to operate in the proposed directions. Here the appropriate sound will be played for every hit detected.

Augmented Reality Musical Instruments system makes the work easier and simple by not manually operating through the devices such as mouse or keyboard but by using the simple web camera and operating the devices through the gesture control mechanism. That is the hand gestures for playing a particular musical instrument performed on a template in the view of the web camera are captured and the system operates on it to ouput sound. This simple method of controlling systems by gestures is becoming a trend these days.

Also non-gesture recognition systems increase the cost and unnecessary hardwareís while this system is less costly and also efficient to use.

Different applications where Augmented Reality Musical Instruments system can be used in real world are as given below:

- When user wants to play a musical instrument to pass his time.
- When a user wants to take up learning a musical instrument as his hobby but is not ready to invest such a huge amount at the very initial stages. Such users can use our system in the initial stages and then if they develop an interest and have learned to play the instrument well and wish to pursue it further might take a decision to buy that musical instrument. Thus our system helps users to take a proper decision.
- When users are travelling to faraway places and the luggage is a restriction, our system can be used to play the desired musical instrument without having to carry it along.

1.3 AIM AND SCOPE

This project presents a technique of image processing for Gesture Recognition Control System, using Gray Scale Histogram of the image. Aim of the system is to recognize the gesture of the user and according to his/her gesture control the features of desktop applications. Gesture is captured using camera, gray scale value of the image is calculated and then the background of the image is suppressed using Gray level Slicing or by setting a threshold value to certain value that it detects only the object which is closer to the camera. And then depending on the captured image application is manipulated i.e. it can be scrolled up or down or can be used for zooming or opening and closing of application.

The things that has to be taken into consideration while developing the proposed system is that the background has to be black/white so it will help in avoiding the confusion/noise while extracting the gesture from image that has been captured. The second thing would be light intensity has to be predefined or light source position has to be pre-defined. These drawbacks can be rectified in the future versions.

In our system we are eliminating the use of wearable like gloves, rings etc. which increases the work output.

Two methods are considered suitable for gesture recognition. The first one is to use vision sensors like cameras to acquire images, which are analyzed to recognize the gestures. The second one is to place inertial sensor beside the camera and extract the motion characters.

1.4 INTENDED AUDIENCE

INTRODUCTION

The intended audience of the system would be the customers who wish to play instruments but prefer a low cost alternative to actually purchasing an actual instrument.

2. REVIEW OF LITERATURE

Augmented reality:

Augmented reality (AR) is a live direct or indirect view of a physical, real-world environment whose elements are augmented (or supplemented) by computer-generated sensory input such as sound, video, graphics or GPS data. It is related to a more general concept called mediated reality, in which a view of reality is modified (possibly even diminished rather than augmented) by a computer. As a result, the technology functions by enhancing one's current perception of reality. By contrast, virtual reality replaces the real world with a simulated one. Augmentation is conventionally in real-time and in semantic context with environmental elements, such as sports scores on TV during a match. With the help of advanced AR technology (e.g. adding computer vision and object recognition) the information about the surrounding real world of the user becomes interactive and digitally manipulable. Artificial information about the environment and its objects can be overlaid on the real world.

Image processing:

In imaging science, image processing is processing of images using mathematical operations by using any form of signal processing for which the input is an image, such as a photograph or video frame; the output of image processing may be either an image or a set of characteristics or parameters related to the image. Image processing is a method to convert an image into digital form and perform some operations on it, in order to get an enhanced image or to extract some useful information from it. Most image-processing techniques involve treating the image as a two-dimensional signal and applying standard signal-processing

techniques to it. Image processing usually refers to digital image processing, but optical and analog image processing also are possible.

Music:

Music is an art form, social activity or cultural activity whose medium is sound and silence. The common elements of music are pitch (which governs melody and harmony), rhythm (and its associated concepts tempo, meter, and articulation), dynamics, and the sonic qualities of timbre and texture (which are sometimes termed the "color" of a musical sound). Different styles or types of music may emphasize, de-emphasize or omit some of these elements. Music is performed with a vast range of instruments and with vocal techniques ranging from singing to rapping, and there are solely instrumental pieces, solely vocal pieces and pieces that combine singing and instruments.

3. REPORT ON PRESENT INVESTIGATION

FIGURE NO. 1 FEATURE RECOGNITION STEPS

Most of the researchers classified feature recognition system into mainly three steps after acquiring the input image from camera. These steps are: Extraction Method, Features estimation and extraction, and Classification or Recognition as illustrated in figure above.

3.1 DESIGN CONSIDERATION

Our implementation methodology uses the input video in two modules as shown in Fig 4.1. The input frames are used to calibrate the markings with the template images stored in the database of the application. The other use of the input frames after the calibration is done is to detect the marker position. Once both the marker position and calibration with the original template is done we can successfully trace which component is the user is trying to play on the paper. The application will further play the sound intended by the user.

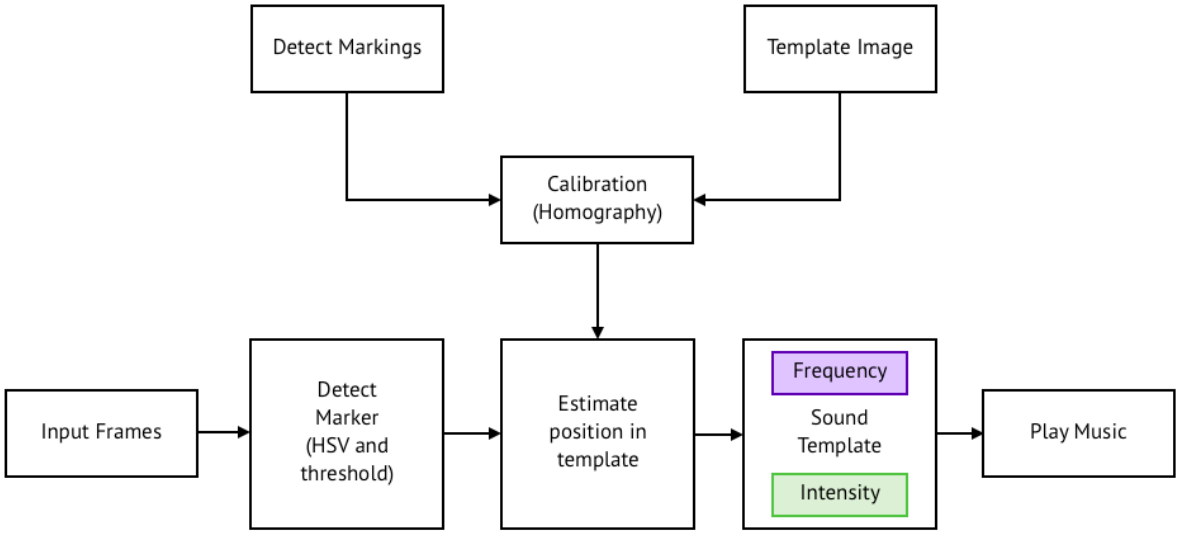


FIGURE NO. 2 PROCESS FLOW

3.2 REQUIREMENTS

3.2.1 FUNCTIONAL REQUIREMENTS

1. Design appropriate sheets that will correspond to different musical instruments. These sheets should be designed in a manner that is easy for the camera to capture, while minimizing the effect of natural occlusions of hand parts while playing the instrument.
2. Developing a program using MATLAB that will perform the following:
   a) Perform calibration to determine the locations of keyboard keys/drums in the real world.
   b) Recognize the gestures/movements of the user and relate them to the playing of the musical instrument.
   c) Output appropriate sound that corresponds to those gestures.

3.2.2 NON-FUNCTIONAL REQUIREMENTS

1. Robustness: Calibration can be made more robust against lighting and background changes.
2. Less Latency: Optimize the program for real-time processing. We observe that most augmented reality products in the market suffer from latency issues. That is, the effect, in this case the music, is not heard as soon as the key press is simulated. To solve this problem, we need to ensure that our program performs in a very efficient manner, and this usually requires a lot of code optimization in terms of speed and memory.
3. Reduced Processing Power: Camera input required will be low resolution thereby increasing the overall speed. Thus the performance of the system will be fairly good in terms of speed and reliability.
4. Scalability: Modularized processes enabling extensibility to new instruments feasible.
5. Usability: The user of this system will find this system very easy to use and highly interactive.

3.3 METHODOLOGY USED

3.3.1 PROPOSED TECHNIQUE

Instead of using traditional key pressing or touch input, we plan to use only laptop camera to capture the user's gestures with a background of a predefined template paper.

The software works in the following way:

1) Calibration of the template.
2) Detecting the marker position.
3) Identifying the hit position.
4) Playing notes corresponding to that position.

3.3.1.1 Calibration of the template

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. Since the page occupies the largest area in the input image, the largest connected component is expected to belong to the page. The detected region is then eroded with a 5×5 mask to remove spurious detections. Then we look for connected components located inside the detected region corresponding to the page.

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. This is done using SIFT algorithm.

3.3.1.2 Detecting the marker position

We have used color object tracking to detect the marker and this is accomplished using Image Acquisition ToolBox provided by MATLAB.

We first define a video object to hold the video captured from our specified webcam at a chosen resolution. Setting it to RGB scale is required. Frame interval should be low

eg. 5 msec. We set a limit for the grabbed frames and start the video. Suppose the colour of the marker is blue. We first subtract the blue components from the grayscale image to track. We then use a median filter to remove the noise. The resulting Grayscale image is converted to binary. We remove all pixels below a certain threshold (say 300px). We label all connected components in the image. We get a set of properties for each labelled part by doing an image Blob Analysis. The properties useful here are those of the Bounding Box and Centroid. Given the areas provided by these properties, we draw a red rectangle around all our components.

3.3.1.3 Hit detection

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 currentm at the end of the queue and pop out the oldest element from the queue. We keep a history of 2 ym 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.

3.3.1.4 Playing notes corresponding to that position

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.

3.3.2 GUI Design

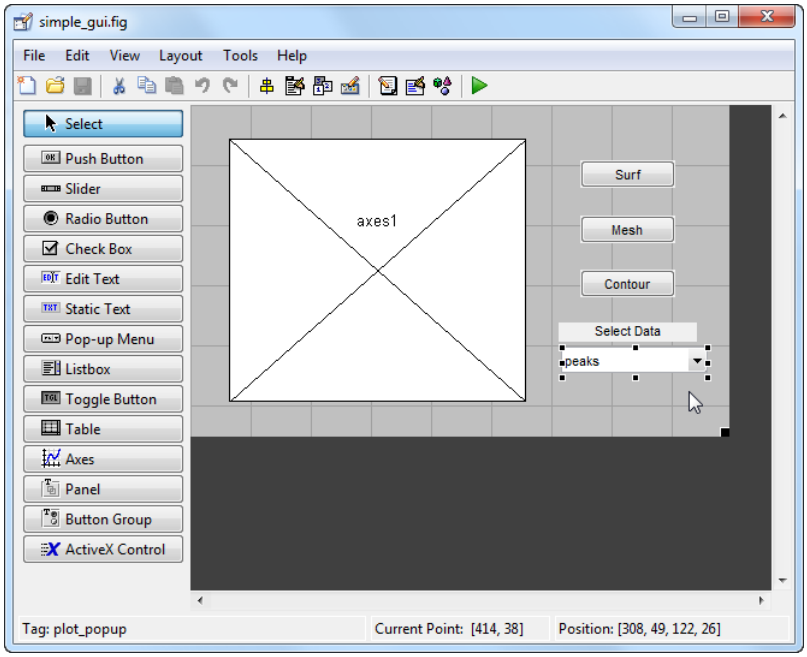
GUIs (also known as graphical user interfaces or UIs) provide point-and-click control of software applications, eliminating the need to learn a language or type commands in order to run the application. In our program we add the GUI element to allow the user to freely enable the hit detection feature whenever he desires. We also let the user decide the color to track from three for the object as well as recalibrate the pattern detected by the software in case of any changes in the environment or the light. For MATLAB, apps are self-contained MATLAB programs with GUI front ends that automate a task or calculation. The GUI typically contains controls such as menus, toolbars, buttons, and sliders. Many MATLAB products, such as Curve Fitting Toolbox, Signal Processing Toolbox, and Control System Toolbox, include apps with custom user.

For our program we make use of a toolbox called GUIDE.

GUIDE

GUIDE (graphical user interface design environment) provides tools for designing user interfaces for custom apps. Using the GUIDE Layout Editor, you can graphically design your UI. GUIDE then automatically generates the MATLAB code for constructing the UI, which you can modify to program the behavior of your app.

GUIDE is a drag-and-drop environment for laying out user interfaces (UIs). You code the interactive behavior of your app separately, in the MATLAB editor. Apps you create using GUIDE can display any type of MATLAB plot. GUIDE also provides various interactive components, including menus, tool bars, and tables. Use this approach to create simple apps that can display any type of plot.



Under the script section of the GUI (Launching the GUIDE toolbox in MATLAB will create two files, .fig file and .m file. The .fig file will contain the layout of the GUI where as the .m file contains the script for the GUI), there are a numerous lines of code automatically generated for each of the element in the layout.

For the video stream:

- Go to the section which resembles the lines “For axes”.
- Set the handles to axes graph so that any graph (in our case the frame stream) will be displayed in the axes graph space.

For the start button:

- Go to the section which resembles the lines “For start_button”.
- Set the action to be performed as the whole program with an exception pointing towards the breakpoint for the program which is checked at every iteration for a very short time, if the interrupt is enabled for the same, the program will stop.

For the stop button:

- Go to the section which resembles the lines “For stop_button”.
- Set the action to be performed as enable the breakpoint interrupt so that the program stops executing.

For the recalibrate button:

- Go to the section which resembles the lines “For recalib_button”.

- Enter the module for VL_SIFT.

3.4 ALGORITHM

Image Acquisition Toolbox simplifies the acquisition process by providing a consistent interface across operating systems, hardware devices, and vendors. The toolbox provides multiple ways to access hardware devices from MATLAB and Simulink: the Image Acquisition Tool, a programmatic interface in MATLAB, and a block for Simulink.

The Image Acquisition app enables you to work with image and video acquisition devices and is well suited for interactive configuration of cameras. One can browse all hardware devices available on the computer, change device settings, select a region of interest (ROI), preview an acquisition, acquire images and video, and record data. A preview window helps verify and optimize your acquisition parameters by automatically reflecting any adjustments made to camera properties in the video stream.

Image acquisition:

Step 1: Install Your Image Acquisition Device

Follow the setup instructions that come with your image acquisition device. Setup typically involves:

- Installing the frame grabber board in your computer.
- Installing any software drivers required by the device. These are supplied by the device vendor.
- Connecting a camera to a connector on the frame grabber board.
- Verifying that the camera is working properly by running the application software that came with the camera and viewing a live video stream.
- Generic Windows image acquisition devices, such as webcams and digital video camcorders, typically do not require the installation of a frame grabber board. You connect these devices directly to your computer via a USB or FireWire port.

- After installing and configuring your image acquisition hardware, start MATLAB® on your computer by double-clicking the icon on your desktop. You do not need to perform any special configuration of MATLAB to perform image acquisition.

Step 2: Retrieve Hardware Information

In this step, you get several pieces of information that the toolbox needs to uniquely identify the image acquisition device you want to access. You use this information when you create an image acquisition object, described in Step 3: Create a Video Input Object.

The following table lists this information. You use the imaqhwinfo function to retrieve each item.

Device Information
Description
Adaptor name

An adaptor is the software that the toolbox uses to communicate with an image acquisition device via its device driver. The toolbox includes adaptors for certain vendors of image acquisition equipment and for particular classes of image acquisition devices. See Determining the Adaptor Name for more information.

Device ID

The device ID is a number that the adaptor assigns to uniquely identify each image acquisition device with which it can communicate. See Determining the Device ID for more information.

Note: Specifying the device ID is optional; the toolbox uses the first available device ID as the default.

Video format

The video format specifies the image resolution (width and height) and other aspects of the video stream. Image acquisition devices typically support multiple video formats. See Determining the Supported Video Formats for more information.

Note: Specifying the video format is optional; the toolbox uses one of the supported formats as the default.

Determining the Adaptor Name

To determine the name of the adaptor, enter the imaqhwinfo function at the MATLAB prompt without any arguments.

```
imaqhwinfo

ans =

    InstalledAdaptors: {'dcam'  'winvideo'}

       MATLABVersion: '7.4 (R2007a)'

         ToolboxName: 'Image Acquisition Toolbox'

      ToolboxVersion: '2.1 (R2007a)'
```

In the data returned by imaqhwinfo, the InstalledAdaptors field lists the adaptors that are available on your computer. In this example, imaqhwinfo found two adaptors available on the computer: 'dcam' and 'winvideo'. The listing on your computer might contain only one adaptor name. Select the adaptor name that provides access to your image acquisition device. For more information, see Determining the Device Adaptor Name.

Determining the Device ID

To find the device ID of a particular image acquisition device, enter the imaqhwinfo function at the MATLAB prompt, specifying the name of the adaptor as the only argument. (You found the adaptor name in the first call to imaqhwinfo, described in Determining the Adaptor Name.) In the data returned, the DeviceIDs field is a cell array containing the device IDs of all the devices accessible through the specified adaptor.

Note This example uses the DCAM adaptor. You should substitute the name of the adaptor you would like to use.

```
info = imaqhwinfo('dcam')

info =

      AdaptorDllName: [1x77 char]

   AdaptorDllVersion: '2.1 (R2007a)'

        AdaptorName: 'dcam'

         DeviceIDs: {[1]}

        DeviceInfo: [1x1 struct]
```

Determining the Supported Video Formats

To determine which video formats an image acquisition device supports, look in the DeviceInfo field of the data returned by imaqhwinfo. The DeviceInfo field is a structure array where each structure provides information about a particular device. To view the device information for a particular device, you can use the device ID as a reference into the structure array. Alternatively, you can view the information for a particular device by calling the imaqhwinfo function, specifying the adaptor name and device ID as arguments.

To get the list of the video formats supported by a device, look at SupportedFormats field in the device information structure. The SupportedFormats field is a cell array of strings where each string is the name of a video format supported by the device. For more information, see Determining Supported Video Formats.

```
dev_info = imaqhwinfo('dcam',1)

dev_info =

        DefaultFormat: 'F7_Y8_1024x768'

        DeviceFileSupported: 0
```

DeviceName: 'XCD-X700 1.05'

DeviceID: 1

VideoInputConstructor: 'videoinput('dcam', 1)'

VideoDeviceConstructor: 'imaq.VideoDevice('dcam', 1)'

SupportedFormats: {'F7_Y8_1024x768' 'Y8_1024x768'}

Step 3: Create a Video Input Object

In this step you create the video input object that the toolbox uses to represent the connection between MATLAB and an image acquisition device. Using the properties of a video input object, you can control many aspects of the image acquisition process. For more information about image acquisition objects, see Creating Image Acquisition Objects.

To create a video input object, use the videoinput function at the MATLAB prompt. The DeviceInfo structure returned by the imaqhwinfo function contains the default videoinput function syntax for a device in the VideoInputConstructor field. For more information the device information structure, see Determining the Supported Video Formats.

The following example creates a video input object for the DCAM adaptor. Substitute the adaptor name of the image acquisition device available on your system.

vid = videoinput('dcam',1,'Y8_1024x768')

The videoinput function accepts three arguments: the adaptor name, device ID, and video format. You retrieved this information in step 2. The adaptor name is the only required argument; the videoinput function can use defaults for the device ID and video format. To determine the default video format, look at the DefaultFormat field in the device information structure. See Determining the Supported Video Formats for more information.

Instead of specifying the video format, you can optionally specify the name of a device configuration file, also known as a camera file. Device configuration files are typically supplied by frame grabber vendors. These files contain all the required configuration settings to use a

particular camera with the device. See Using Device Configuration Files (Camera Files) for more information.

Viewing the Video Input Object Summary

To view a summary of the video input object you just created, enter the variable name vid at the MATLAB command prompt. The summary information displayed shows many of the characteristics of the object, such as the number of frames that will be captured with each trigger, the trigger type, and the current state of the object. You can use video input object properties to control many of these characteristics. See Step 5: Configure Object Properties (Optional) for more information.

vid

Summary of Video Input Object Using 'XCD-X700 1.05'.

Acquisition Source(s): input1 is available.

Acquisition Parameters: 'input1' is the current selected source.

10 frames per trigger using the selected source.

'Y8_1024x768' video data to be logged upon START.

Grabbing first of every 1 frame(s).

Log data to 'memory' on trigger.

Trigger Parameters: 1 'immediate' trigger(s) on START.

Status: Waiting for START.

0 frames acquired since starting.

0 frames available for GETDATA.

Step 4: Preview the Video Stream (Optional)

After you create the video input object, MATLAB is able to access the image acquisition device and is ready to acquire data. However, before you begin, you might want to see a preview of the video stream to make sure that the image is satisfactory. For example, you might want to change the position of the camera, change the lighting, correct the focus, or make some other change to your image acquisition setup.

preview(vid)

The preview function opens a Video Preview figure window on your screen containing the live video stream. To stop the stream of live video, you can call the stoppreview function. To restart the preview stream, call preview again on the same video input object.

While a preview window is open, the video input object sets the value of the Previewing property to 'on'. If you change characteristics of the image by setting image acquisition object properties, the image displayed in the preview window reflects the change.

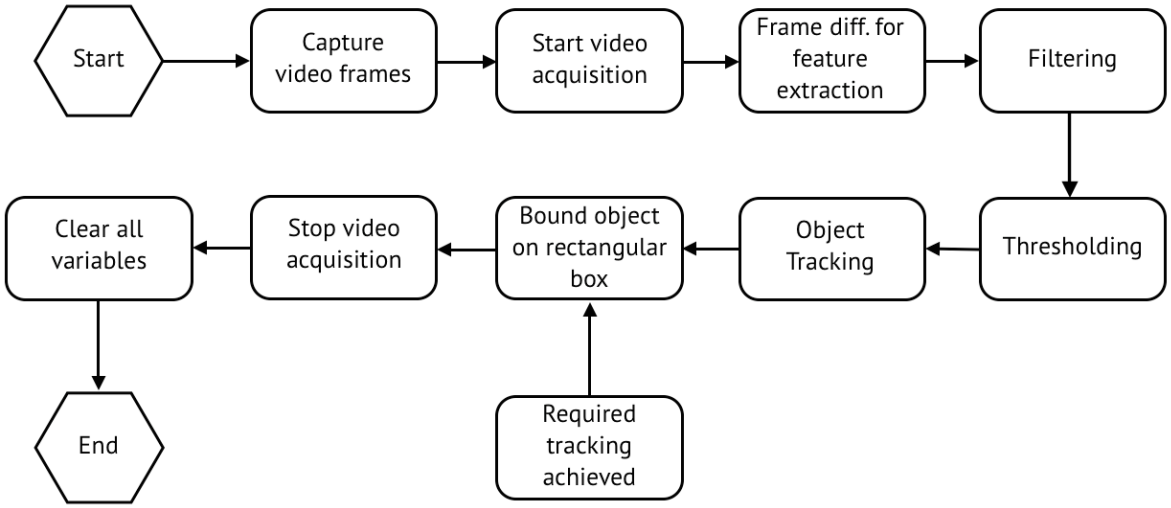
To close the Video Preview window, click the Close button in the title bar or use the closepreview function, specifying the video input object as an argument.

closepreview(vid)

Calling closepreview without any arguments closes all open Video Preview windows.

- Colour tracking algorithm

Several general purpose techniques and algorithms have been developed for object tracking.



As we have already seen that during the tracking of the object light illumination acts as noise,

also we need that the time required for the processing of the image should be as low as possible. To perform video tracking an algorithm analyzes video frames and outputs the movement of targets between the frames. There are a variety of algorithms each having strength and weakness. May, the proposed algorithm discussed in this paper will be helpful in developing better and efficient algorithms in the field of tracking. A basic flow chart diagram for the proposed algorithm is shown above.

The blocks present in the flowchart are being explained below in the following steps:

Step1: Capture the video frames using the video input function.

Step2: Set the properties of video object.

Step3: Start the video acquisition.

Step4: Set a loop that starts after 50 frames of acquisition. This loop contains the following steps:

1. Get the snapshot of the current frame.
2. Now to track the red objects in real time we have to subtract the red component from the gray scale image to extract the red components in the image.
3. Use a median filter to filter out noise.
4. Convert the resulting gray scale image into a binary image.
5. Remove all those pixels which are less than 300 pixels.
6. Label all the connected components in the image to perform image blob analysis, here we get a set of properties for each labelled region.
7. Display the image.
8. Again a loop is used to bound the red objects in a rectangular box.

Step5: Stop the video acquisition.

Step6: Flush all the image data stored in the memory buffer.

Step7 Clear all the variables**.**

3.5 TECHNIQUES AND TECHNOLOGIES USED

3.5.1 vl_sift function

The Scale-Invariant Feature Transform (SIFT) bundles a feature detector and a feature descriptor. The detector extracts from an image a number of frames (attributed regions) in a way which is consistent with (some) variations of the illumination, viewpoint and other

viewing conditions. The descriptor associates to the regions a signature which identifies their appearance compactly and robustly.

Usage:

1. Both the detector and descriptor are accessible by the vl_sift MATLAB command (there is a similar command line utility). We load a test image:

```
I = vl_impattern('roofs1') ;

        image(I) ;
```

2. We compute SIFT frames (keypoints) and descriptors by:

```
[f,d] = vl_sift(I) ;
```

3. For finding similar regions between two images, vl_ubcmatch implements a basic matching algorithm. Let Ia and Ib be images of the same object or scene. We extract and match the descriptors by:

```
[fa, da] = vl_sift(Ia) ;

        [fb, db] = vl_sift(Ib) ;

        [matches, scores] = vl_ubcmatch(da, db) ;
```

4. We run the detector with peak threshold peak_thresh by:

```
f = vl_sift(I, 'PeakThresh', peak_thresh) ;
```

obtaining fewer features as peak_thresh is increased.

5. We run the detector with edge threshold edge_thresh by:

```
f = vl_sift(I, 'edgethresh', edge_thresh) ;
```

obtaining more features as edge_thresh is increased.

Feature recognition using SIFT:

The SIFT approach, for image feature generation, takes an image and transforms it into a "large collection of local feature vectors". To aid the extraction of these features the SIFT algorithm applies a 4 stage filtering approach:

1. **Scale-Space Extrema Detection**

   This stage of the filtering attempts to identify those locations and scales that are identifiable from different views of the same object. This can be efficiently achieved using a "scale space" function. Further it has been shown under reasonable assumptions it must be based on the Gaussian function. The scale space is defined by the function:

   $L(x, y, \sigma) = G(x, y, \sigma) * I(x, y)$

   Where * is the convolution operator, $G(x, y, \sigma)$ is a variable-scale Gaussian and $I(x, y)$ is the input image.

   Various techniques can then be used to detect stable keypoint locations in the scale-space. Difference of Gaussians is one such technique, locating scale-space extreme, $D(x, y, \sigma)$ by computing the difference between two images, one with scale k times the other. $D(x, y, \sigma)$ is then given by:

   $D(x, y, \sigma) = L(x, y, k\sigma) - L(x, y, \sigma)$

   To detect the local maxima and minima of $D(x, y, \sigma)$ each point is compared with its 8 neighbours at the same scale, and its 9 neighbours up and down one scale. If this value is the minimum or maximum of all these points then this point is an extrema.

2. **Keypoint Localistaion**

   This stage attempts to eliminate more points from the list of keypoints by finding those that have low contrast or are poorly localised on an edge. This is achieved by calculating the Laplacian value for each keypoint found in stage 1. The location of extremum, **z**, is given by:

$$\mathbf{z} = -\frac{\partial^2 D}{\partial \mathbf{x}^2}^{-1} \frac{\partial D}{\partial \mathbf{x}}$$

If the function value at **z** is below a threshold value then this point is excluded. This removes extrema with low contrast. To eliminate extrema based on poor localisation it is noted that in these cases there is a large principle curvature across the edge but a small curvature in the perpendicular direction in the defference of Gaussian function. If this difference is below the ratio of largest to smallest eigenvector, from the 2x2 Hessian matrix at the location and scale of the keypoint, the keypoint is rejected.

3. Orientation Assignment

This step aims to assign a consistent orientation to the keypoints based on local image properties. The keypoint descriptor, described below, can then be represented relative to this orientation, achieving invariance to rotation. The approach taken to find an orientation is:

- Use the keypoints scale to select the Gaussian smoothed image L, from above
- Compute gradient magnitude, *m*

$$m(x, y) = \sqrt{(L(x+1, y) - L(x-1, y))^2 + (L(x, y+1) - L(x, y-1))^2}$$

- Compute orientation, θ

$$\theta(x, y) = \tan^{-1}((L(x, y+1) - L(x, y-1))/(L(x+1, y) - L(x-1, y)))$$

- Form an orientation histogram from gradient orientations of sample points
- Locate the highest peak in the histogram. Use this peak and any other local peak within 80% of the height of this peak to create a keypoint with that orientation
- Some points will be assigned multiple orientations
- Fit a parabola to the 3 histogram values closest to each peak to interpolate the peaks position

4. Keypoint Descriptor

The local gradient data, used above, is also used to create keypoint descriptors. The gradient information is rotated to line up with the orientation of the keypoint and then weighted by a Gaussian with variance of 1.5 * keypoint scale. This data is then used to create a set of histograms over a window centred on the keypoint.

Keypoint descriptors typically uses a set of 16 histograms, aligned in a 4x4 grid, each with 8 orientation bins, one for each of the main compass directions and one for each of the mid-points of these directions. This results in a feature vector containing 128 elements.

These resulting vectors are know as SIFT keys and are used in a nearest-neighbours approach to identify possible objects in an image. Collections of keys that agree on a possible model are identified, when 3 or more keys agree on the model parameters this model is evident in the image with high probability. Due to the large number of SIFT keys in an image of an object, typically a 500x500 pixel image will generate in the region of 2000 features, substantial levels of occlusion are possible while the image is still recognized by this technique.

3.5 DIAGRAMS

3.5.1 FLOW CHART

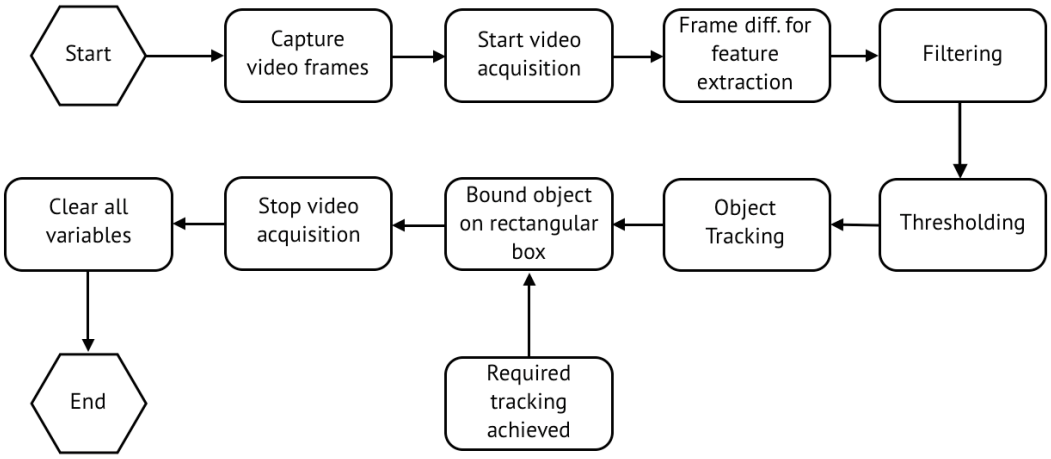


FIGURE NO. 7 FLOW CHART

3.5.2 SEQUENCE DIAGRAM

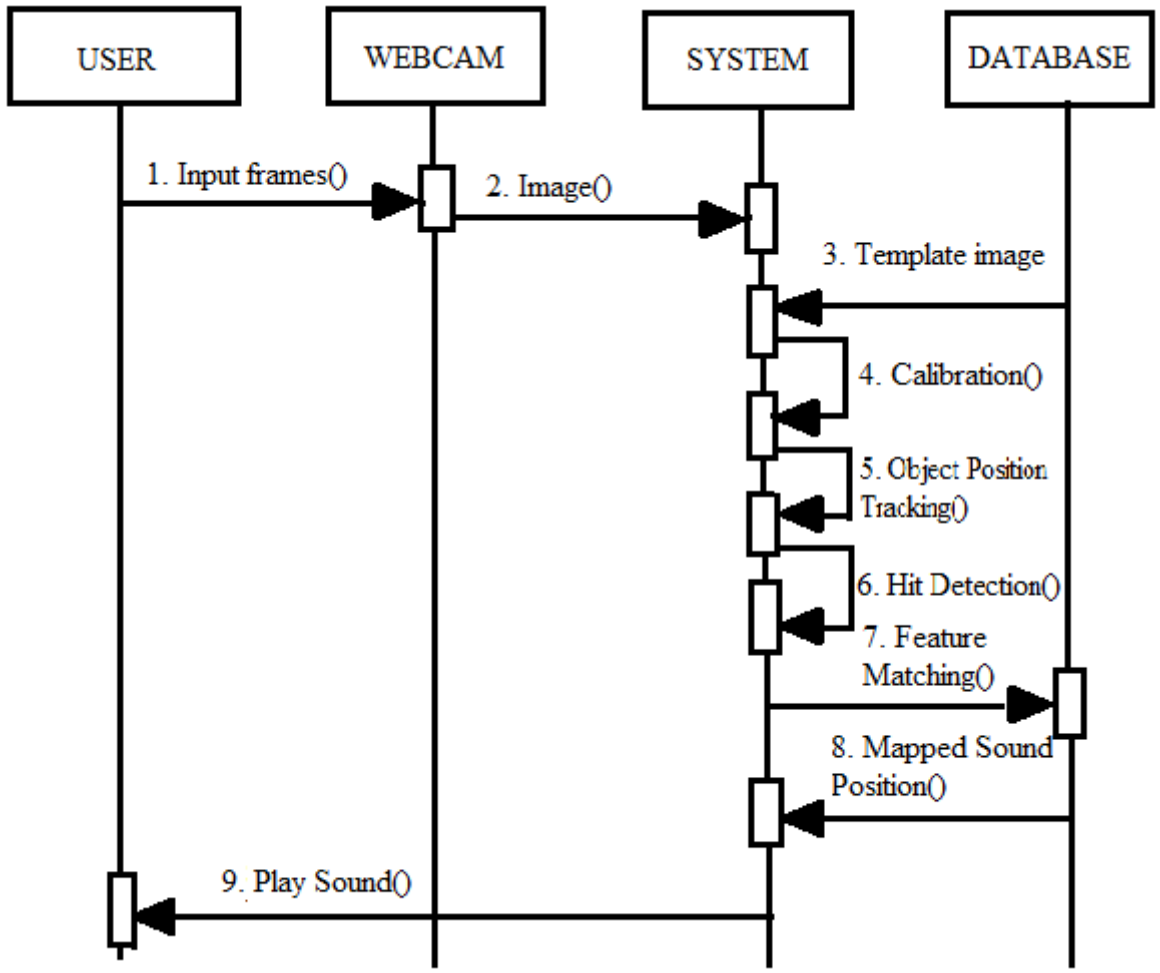


FIGURE NO. 8 SEQUENCE DIAGRAM

3.5.3 USE-CASE DIAGRAM

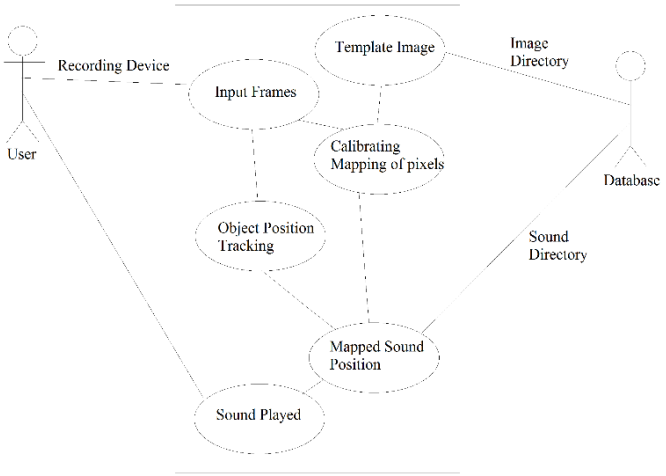


FIGURE NO. 9 USE CASE DIAGRAM

3.5.3 DATA FLOW DIAGRAMS

3.5.3.1 Context level DFD

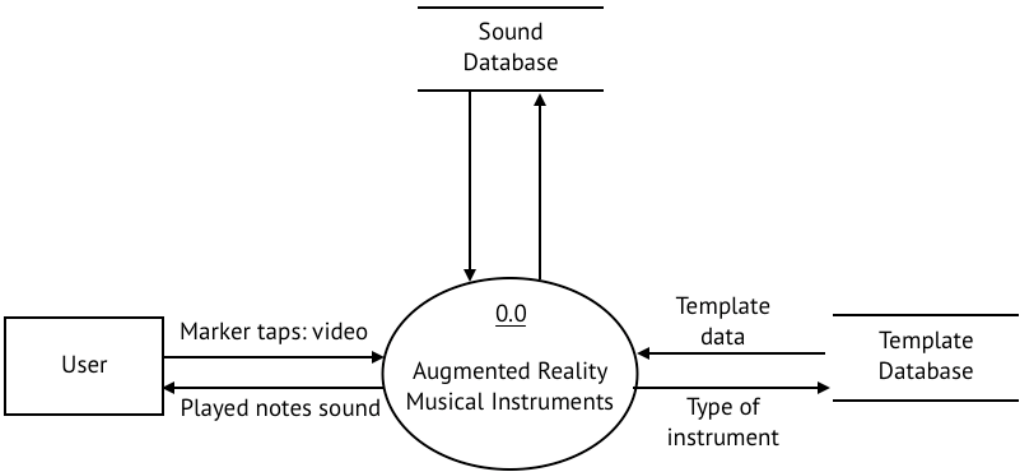


FIGURE NO. 10 CONTEXT LEVEL DFD

3.5.3.2 Level 0

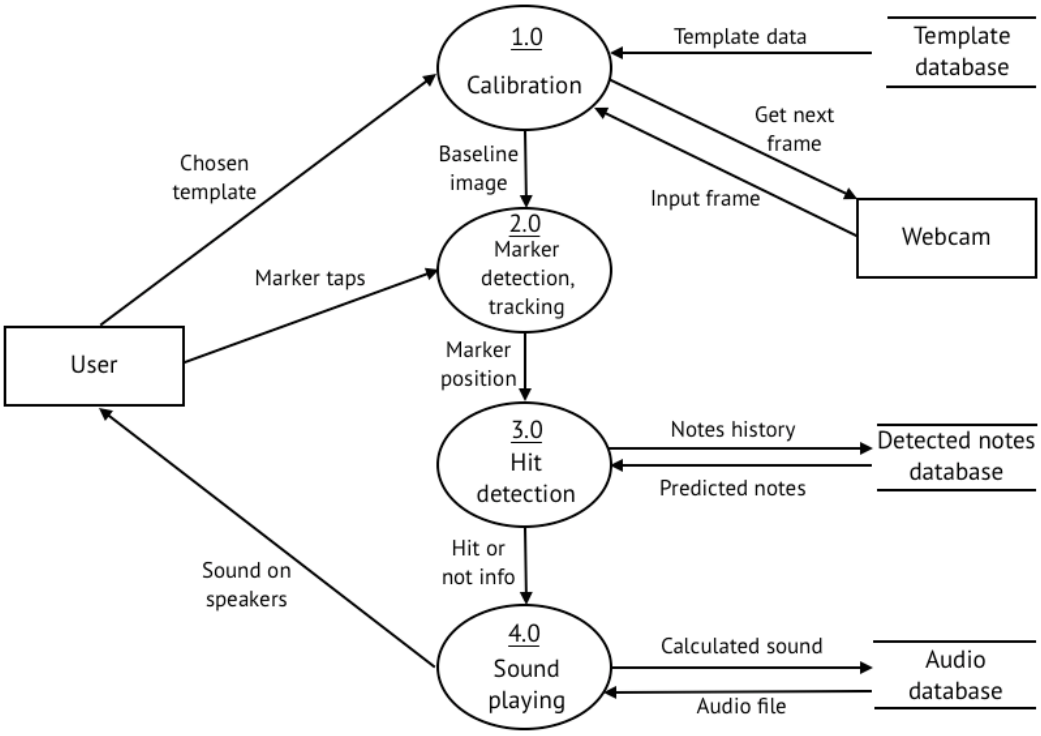


FIGURE NO. 11 LEVEL 0 DFD

3.5.3.3 Level 1.1

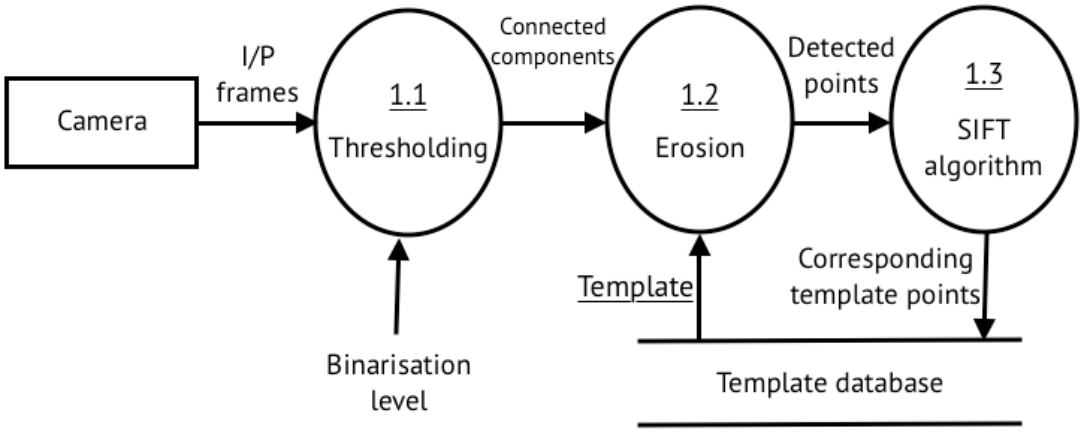


FIGURE NO. 12 LEVEL 1.1 DFD

3.5.3.4 Level 1.2

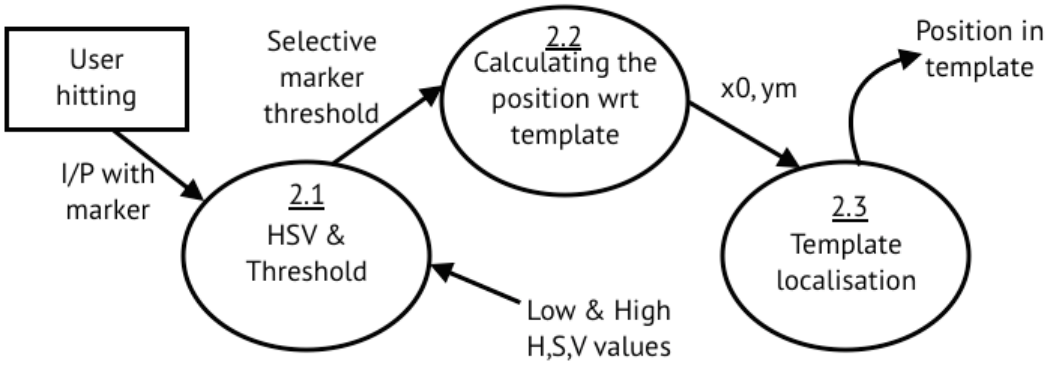


FIGURE NO. 13 LEVEL 1.2 DFD

3.5.3.5 Level 1.3

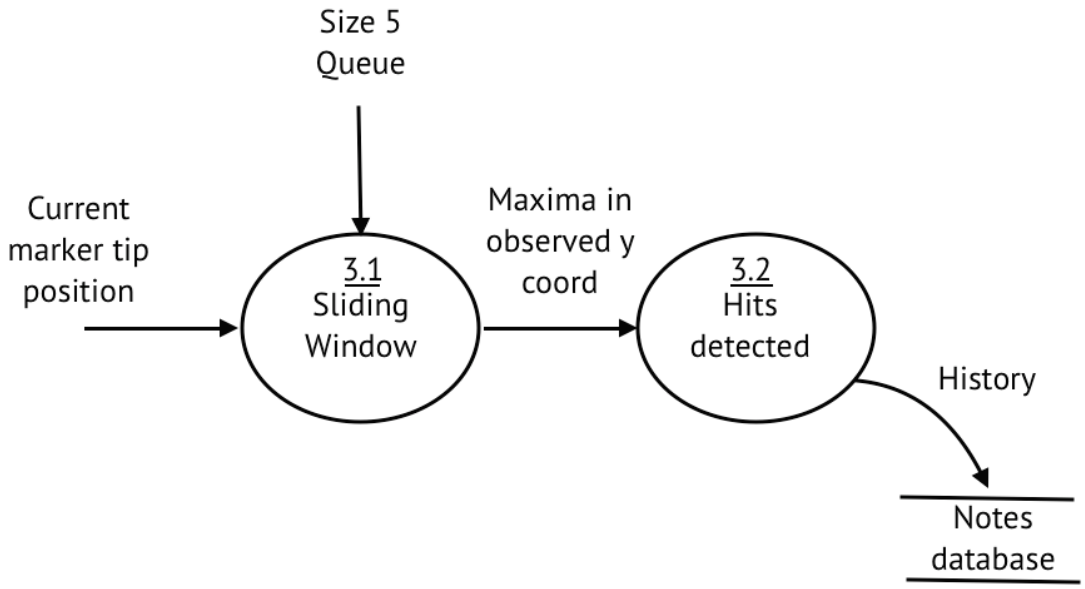


FIGURE NO. 14 LEVEL 1.3 DFD

3.5.3.6 Level 1.4

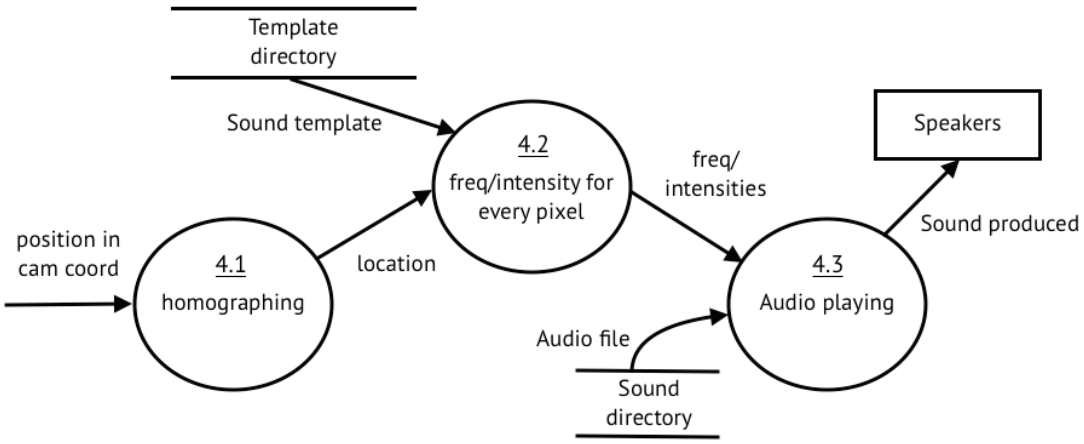


FIGURE NO. 15 LEVEL 1.4 DFD

3.6 HARDWARE REQUIREMENTS

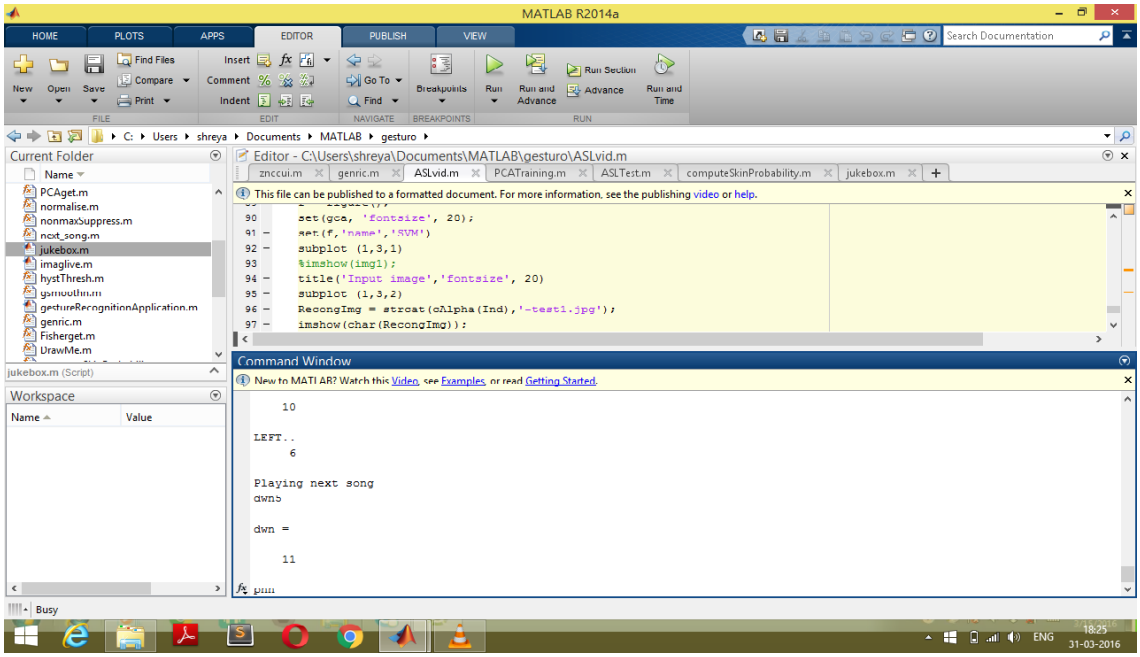
- Laptop (has an inbuilt webcam)
- Intel i3 processor (2.2 GHz).
- 2 GB RAM.

3.7 SOFTWARE REQUIREMENTS

- Camera drivers must be installed.
- Operating System: Windows XP or Above.
- Matlab2014a
- vl_feat package installed within Matlab.

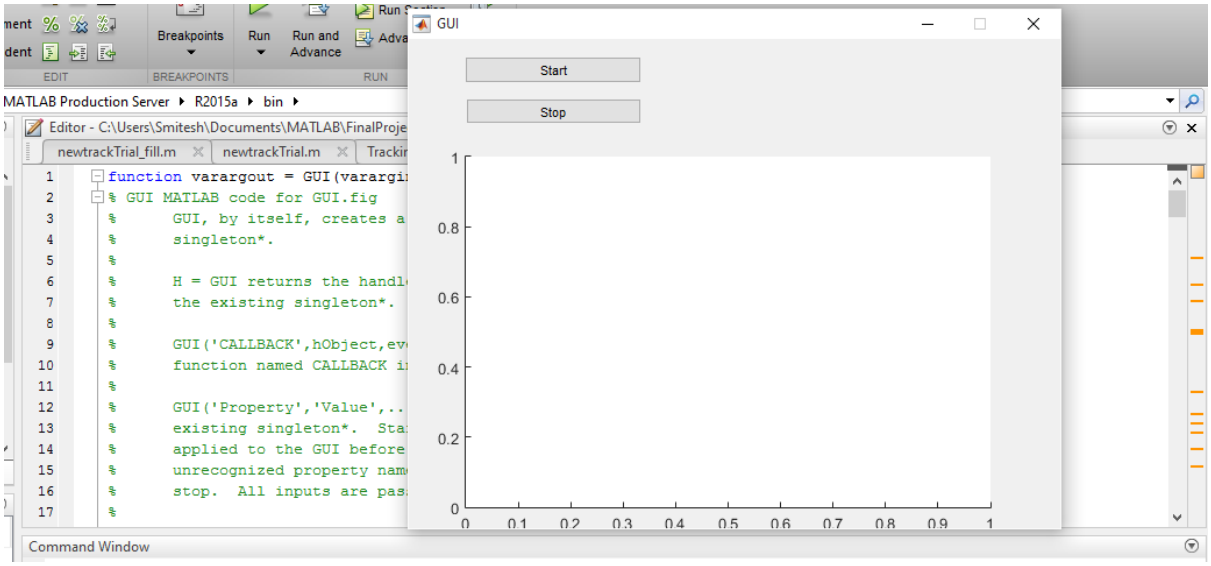
4. RESULTS AND DISCUSSIONS

1. MATLAB UI



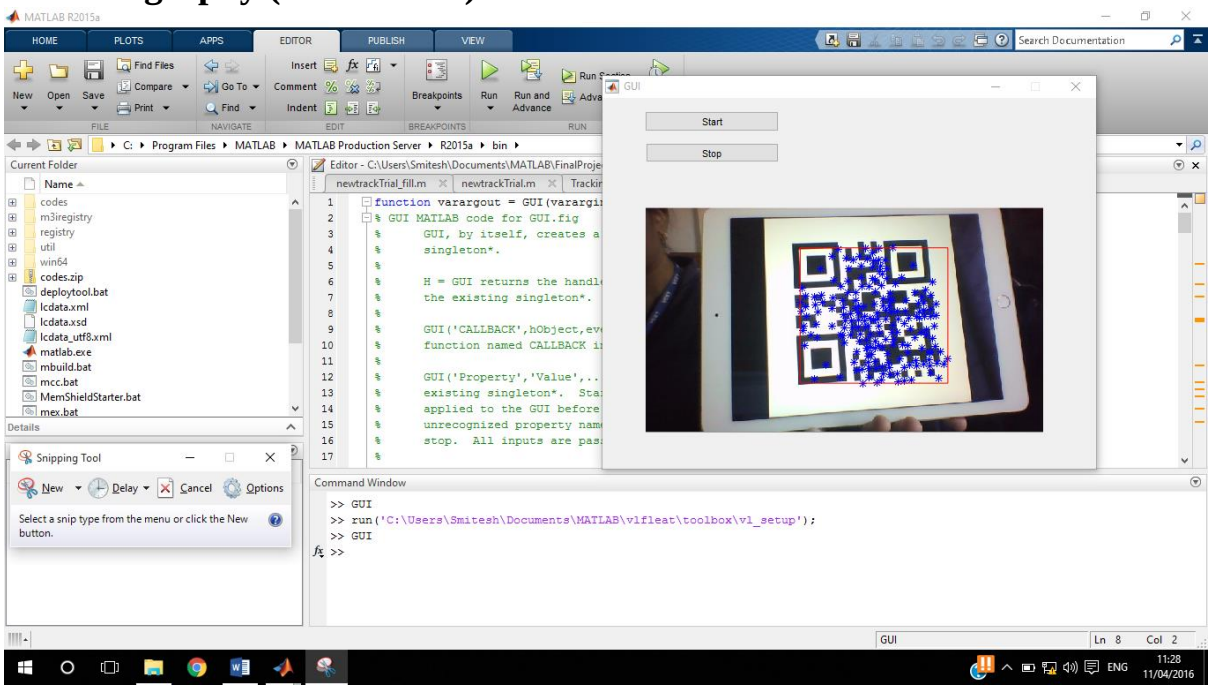
The project starts execution from this page while we click on the run tab. Newtrack.m file is executed first.

2. GUI Layout (Startup).



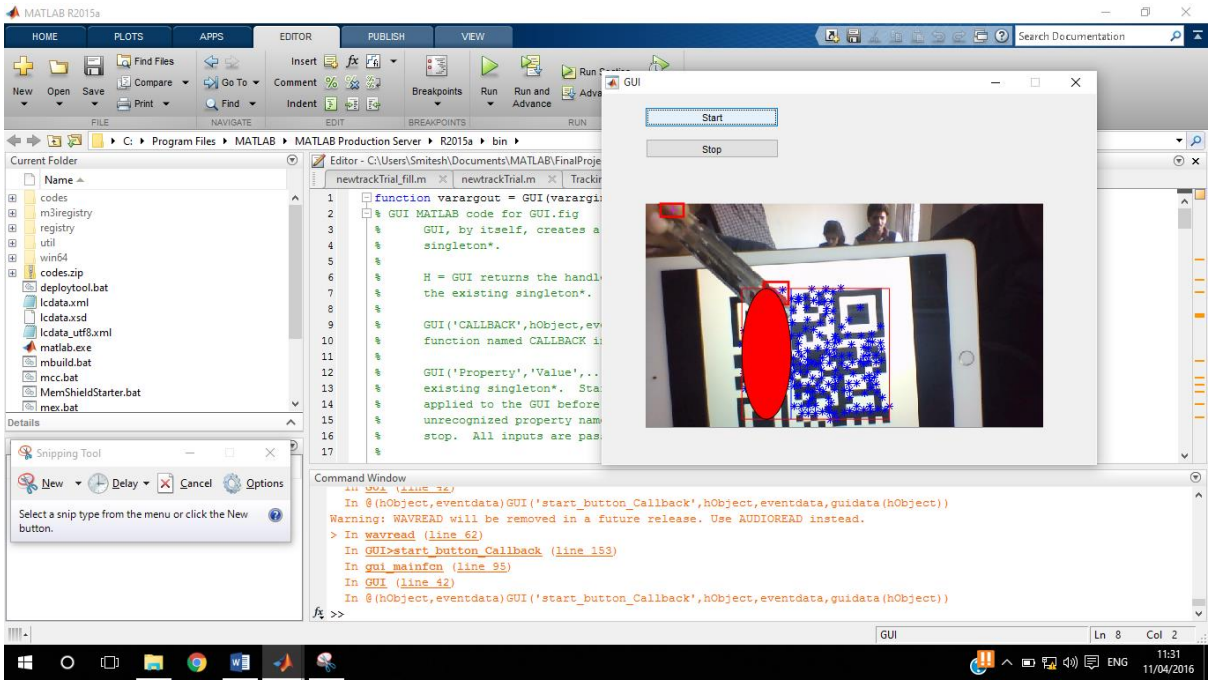
The GUI starts up with the empty plot. This is the initial state. The GUI features the controls start and stop and displays the plot plane for the SIFT coordinates.

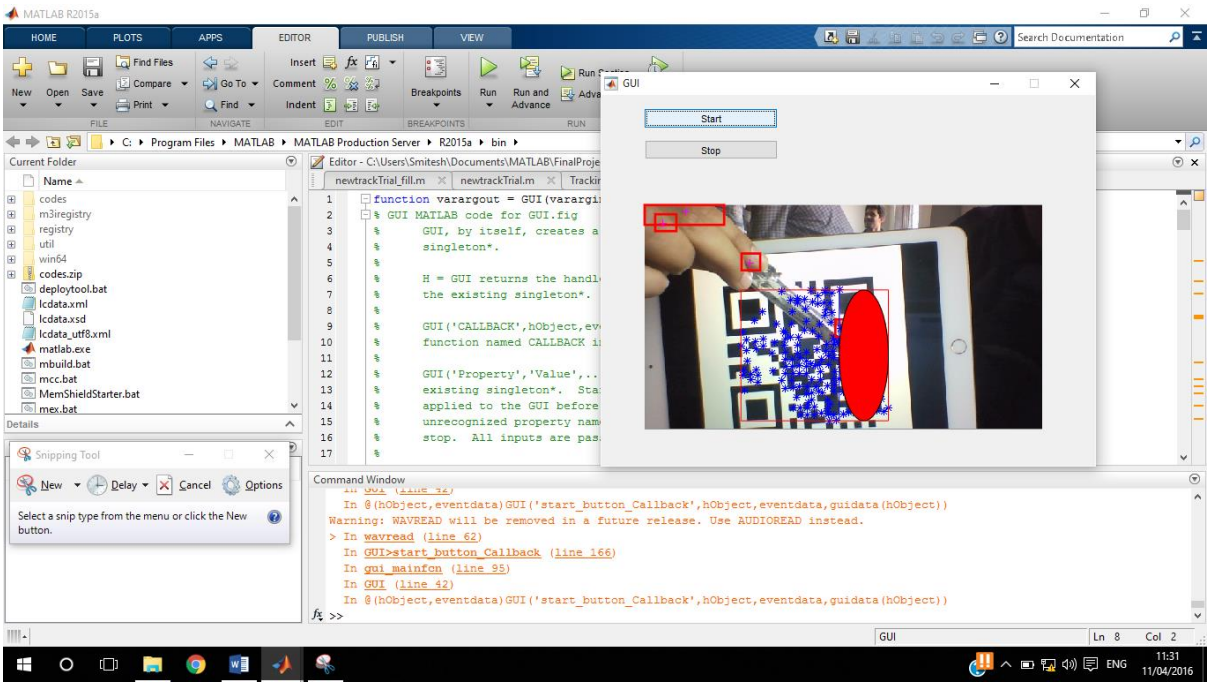
3. Homography (Calibration)



On clicking the start button, the webcam begins capturing video frames. The template held in front of screen is detected, plotting the calibrated bounding box. This will be the play area.

4. Hit Detection:





On tapping the red pointed marker over every third area, a corresponding note is played according to the area in which the marker is placed.

5. CONCLUSIONS AND FUTURE SCOPE

5.1 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.

5.2 FUTURE SCOPE

The present application though seems to be reasonably feasible and user friendly but is somewhat less robust in recognition phase. An attempt can be made to make the calibration more robust against lighting and background changes. 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. Connecting an android device with the laptop via Bluetooth and playing the music in real- time on the phone while the user is playing in front of the laptop. Normal hit speeds can be detected easily, however the performance deteriorates as the speed increases.

6. APPENDIX

6.1 MATLAB 2015a

Matlab developed by Mathworks is a high level language and an interactive environment for numerical computation of data and can be used for developing algorithms, analyzing, visualizing data. The data can be imported int6o Matlab from files, other applications or very any external device for performing the desired computational analysis. Matlab Language supports vector operations which are the basics of engineering and scientific problems.

6.2 vl_feat Package

The VLFeat open source library implements popular computer vision algorithms specializing in image understanding and local features extraction and matching. Algorithms include Fisher Vector, VLAD, SIFT, MSER, k-means, hierarchical k-means, agglomerative information bottleneck, SLIC superpixels, quick shift superpixels, large scale SVM training, and many others. It is written in C for efficiency and compatibility, with interfaces in MATLAB for ease of use, and detailed documentation throughout. It supports Windows, Mac OS X, and Linux.

6.3 SIFT Algorithm

Scale Invariant Feature Transforms. SIFT image features provide a set of features of an object that are not affected by many of the complications experienced in other methods, such as object scaling and rotation.

6.4 Image Acquisition Toolbox

Image Acquisition Toolbox™ enables you to acquire images and video from cameras and frame grabbers directly into MATLAB® and Simulink®. You can detect hardware automatically and configure hardware properties. Advanced workflows let you trigger acquisition while processing in-the-loop, perform background acquisition, and synchronize sampling across several multimodal devices. With support for multiple hardware vendors and industry standards, you can use imaging devices ranging from inexpensive Web cameras to high-end scientific and industrial devices that meet low-light, high-speed, and other challenging requirements.

7. LITERATURE CITED

[1]*MohamedAlsheakhali, "Hand Gesture Recognition System"* **,***The Islamic University of Gaza Strip***,** *Year2011***.**

[2] *RustamIgorevich, "Hand gesture recognition algorithm based on Grayscale histogram of the image", Korea, Year 2010.*

[3]*P.Viola and M.Jones, "Rapid object detection using a boosted cascade of simple features", Year 2001.*

[4] *Rafiqul Khan and Noor Ibraheem, "Hand gesture recognition : Literature Review", Year 2012.*

[5*] https://martin-thoma.com/zero-mean-normalized-cross-correlation/*

[6] *Hamid A. Jalab and Herman .K. Omer, "Human Computer Interface Using Hand Gesture Recognition Based On Neural Network", Year 2015, IEEE*

[7] http://clickdamage.com/sourcecode/index.php

8. ACKNOWLEDGEMENT

It is nearly impossible to thank everyone who ever helped us in the process of taking Data solution to live running model. This book is a direct result of the help of all people who directly or indirectly helped us realizing our goal.

We are heartily thankful to our internal guide Mrs. Sangita Chaudhari for noble guidance and moral support. Their valuable suggestion timely advise, inspired us towards sustain efforts for our project work.

Special thanks to Prof. Swapna Borde, HOD of Information Technology Department for their encouragement and motivation. Finally we are thankful to our principal Dr. Ashok V. Bhonsale and entire Computer Department for providing us excellent infrastructure for carrying our project work.

Ms. Surabhi D. Pawar
Mr. Smitesh Sawant
Ms. Prateeksha S. Singh