International Journal of Computer Architecture and Mobility (ISSN 2319-9229) Volume 3 -Issue 4, June 2015

Drishti – Automated Machine Interface

Biswarup Nandi[#1], Souvik Sur[*2], Upasana Roy Chowdhury[#3],Romit Beed[#4]

***Abstract*— The project "DRISHTI" has been developed to provide computer access, without requiring any hardware interface. The system tracks the computer user's movements with a webcam and translates them into the movements of the mouse pointer on the screen. Portions such as the tip of the nose or finger can be tracked with high rate of success. This is based on cropping a template of the tracked feature from the current image and testing where this template is in the subsequent frame. The location of the highest match is interpreted as the new location of the feature in the subsequent image and the next image frame is then loaded for further processing.**

***Keywords*— Tracking of webcam data, Extracting templates, Matching of templates, Returning the current position of the sample template, Controlling the cursor**

I. Introduction

Computer vision is the science and technology of machines that see. As a scientific discipline, computer vision is concerned with the theory behind artificial systems that extract information from images. The image data can take many forms, such as video sequences, views from multiple cameras, or multi-dimensional data from a medical scanner.

As a technological discipline, computer vision seeks to apply its theories and models to the construction of computer vision systems. Examples of applications of computer vision include systems for:

- Controlling processes (e.g., an industrial robot or an autonomous vehicle).
- Detecting events (e.g., for visual surveillance or people counting).
- Organizing information (e.g., for indexing databases of images and image sequences).
- Modeling objects or environments (e.g., industrial inspection, medical image analysis or topographical modeling).
- Interaction (e.g., as the input to a device for computer-human interaction).

II. Human Computer Interface

A basic goal of HCI is to improve the interactions between users and computers by making computers more usable and receptive to the user's needs. Specifically, HCI is concerned with:

- Methodologies and processes for designing interfaces (i.e., given a task and a class of users, design the best possible interface within given constraints, optimizing for a desired property such as learnability or efficiency of use)
- Methods for implementing interfaces (e.g. software toolkits and libraries; efficient algorithms)
- Techniques for evaluating and comparing interfaces
- Developing new interfaces and interaction techniques
- Developing descriptive and predictive models and theories of interaction

A long term goal of HCI is to design systems that minimize the barrier between the human's cognitive model of what they want to accomplish and the computer's understanding of the user's task.

III. Approach

The system can be broken into two main sub systems - Template matching and Tracking. Template matching is a technique in digital image processing for finding small parts of an image which match a template image. It can be used in manufacturing as a part of quality control, a way to navigate a mobile robot, or as a way to detect edges in images. Tracking is to track moving objects through a sequence of images.

A. Template Matching

1) Feature-based approach: If the template image has strong features, a feature-based approach may be considered; the approach may prove further useful if the match in the search image might be transformed in some fashion. Since this approach does not consider the entirety of the template image, it can be more computationally efficient when working with source images of larger resolution, as the alternative approach, template-based, may require searching potentially large amounts of points in order to determine the best matching location.

2) Template-based approach: For templates without strong features, or for when the bulk of the template image constitutes the matching image, a template-based approach may be effective. As aforementioned, since template-based template matching may potentially require sampling of a large number of points, it is possible to reduce the number of sampling points by reducing the resolution of the search and template images by the same factor and performing the operation on the resultant downsized images (multiresolution, or pyramid, image processing), providing a search window of data points within the search image so that the template does not have to search every viable data point, or a combination of both.

3) Motion tracking and occlusion handling: In instances where the template may not provide a direct match, it may be useful to implement the use of eigenspaces – templates that detail the matching object under a number of different conditions, such as varying perspectives, illuminations, color contrasts, or acceptable matching object "poses". For example, if the user was looking for a face, the eigenspaces may consist of images (templates) of faces in different positions to the camera, in different lighting conditions, or with different expressions.

It is also possible for the matching image to be obscured, or occluded by an object; in these cases, it is unreasonable to provide a multitude of templates to cover each possible occlusion. For example, the search image may be a playing card, and in some of the search images, the card is obscured by the fingers of someone holding the card, or by another card on top of it, or any object in front of the camera for that matter. In cases where the object is malleable or poseable, motion also becomes a problem, and problems involving both motion and occlusion become ambiguous. In these cases, one possible solution is to divide the template image into multiple sub-images and perform matching on each subdivision.

4) Template-based matching and convolution: A basic method of template matching uses a convolution mask (template), tailored to a specific feature of the search image, which is to be detected. This technique can be easily performed on grey images or edge images. The convolution output will be highest at places where the image structure matches the mask structure, where large image values get multiplied by large mask values.

This method is normally implemented by first picking out a part of the search image to use as a template: The search image is named as S(x, y), where (x, y) represent the coordinates of each pixel in the search image. The template is named as $T(x_t, y_t)$, where (x_t, y_t) represent the coordinates of each pixel in the template. The centre (or the origin) of the template $T(x_t, y_t)$ is then simply moved over each (x, y) point in the search image and the sum of products between the coefficients in S(x, y) and $T(x_t, y_t)$ are calculated over the whole area spanned by the template. As all possible positions of the template with respect to the search image are considered, the position with the highest score is the best position. This method is sometimes referred to as 'Linear Spatial Filtering' and the template is called a filter mask.

For example, one way to handle translation problems on images, using template matching is to compare the intensities of the pixels, using the SAD (Sum of absolute differences) measure[1].

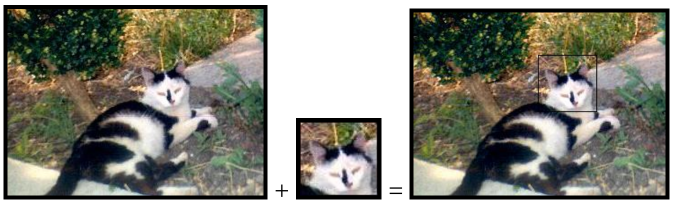
A pixel in the search image with coordinates (x_s, y_s) has intensity $I_s(x_s, y_s)$ and a pixel in the template with coordinates (x_t, y_t) has intensity $I_t(x_t, y_t)$. Thus the absolute difference in the pixel intensities is defined as $\mathrm{Diff}(x_s, y_s, x_t, y_t) = | I_s(x_s, y_s) – I_t(x_t, y_t)$

$$SAD(x,y) = \sum_{i=0}^{T_{\text{rows}}} \sum_{j=0}^{T_{\text{cols}}} \mathrm{Diff}(x+i, y+j, i, j)$$

The mathematical representation of the idea about looping through the pixels in the search image as the origin of the template is translated at every pixel and SAD measure calculation is the following:

$$\sum_{x=0}^{S_{\text{rows}}} \sum_{y=0}^{S_{\text{cols}}} SAD(x,y)$$

S_{rows} and S_{cols} denote the rows and the columns of the search image and T_{rows} and T_{cols} denote the rows and the columns of the template image, respectively. In this method the lowest SAD score gives the estimate for the best position of template within the search image. The method is simple to implement and understand, but it is one of the slowest methods. Example



(a)The first image of the object (b)The second image (c)The template matched in the object

B. Extraction and Tracking

The proposed motion tracking system consists of four main modules

1) Tracking of webcam data: This module deals with webcam interface. In this module, the data is acquired and the webcam is interrupted accordingly when the frames are required for further processing.

2) Extracting templates: This module serves to detect the original template, within the viewing area of the camera in the working space, provided by the user or automatically generated. The input is a sequence of images captured by the camera, and the output states whether or not any template has been detected and stores it as the original template.

3) Template matching: This module serves to detect the original template, within the viewing area of the camera in the working space, provided by the user or automatically generated. The input is a sequence of images captured by the camera, and the output states whether or not any template has been detected and stores it as the original template.

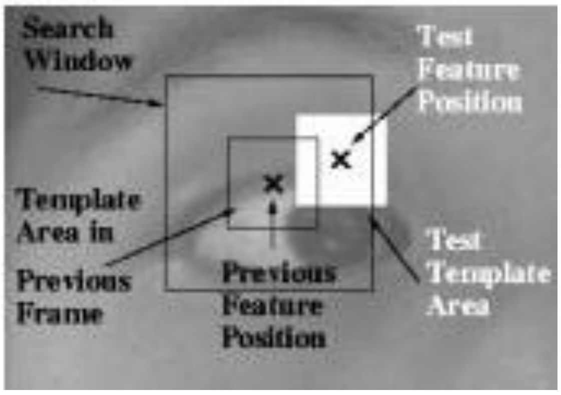


Fig. 1 Template matching using Neighbourhood Window

4) Motion template matching and tracking: The purpose of this module is to find the new location of the moving template within the viewing area of the camera in the workspace. This module is the most critical part of the motion tracking system and its performance greatly affects the accuracy of the final output of the motion tracking system. For the motion tracking system, as a real-time system, tracking speed is crucial. Therefore, the motion template matching methods that are implemented must be fast and efficient. Two alternative motion template matching methods, the neighbourhood window method [2] and the motion window method, have been implemented. The motion window method is activated in cases where the neighbourhood window method fails to find the moving object. Both methods are fast and accurate, and have been implemented using parallel computing techniques to ensure that the computation time is acceptable. In the following two subsections the concepts and aspects of the two alternative motion template matching methods are presented.

Neighbourhood Window Method

Once the motion template is created, the motion tracking system begins motion template matching and tracking. In order to have a good template matching speed, the algorithm does not search the whole image. Rather searches the area of the image that most likely has the new position of the moving object. This area is located in the surrounding neighbourhood of the moving object’s previous position; this area is called the neighbourhood window (see Figure 13). The correlation score is calculated by computing the sum of the absolute difference of pixels between the motion template and the neighbourhood window. The size of the neighbourhood window can be varied for different tracking systems.

The following are the general processing steps of the neighbourhood window method:-

1. Choose the neighbourhood window size, i.e. width = template_width + 4 and length = template_length + 6, and expand the motion template evenly on each side.

2. Compute the sum of pixel correlation values between the motion template and the matching region in the new image. The computed sum value is the correlation score of this region.

3. Repeat step 3, but move the matching region one pixel towards the right (if it is at the end of the row, move to the next row) until the whole neighbourhood window has been processed.4. Compare all the correlation scores and save the best correlation score of that block, and the position of that matching region is the new position of the moving object.

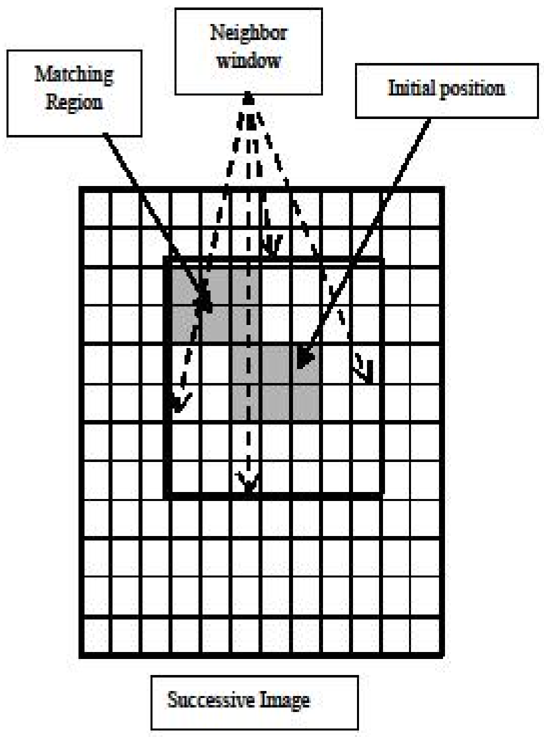


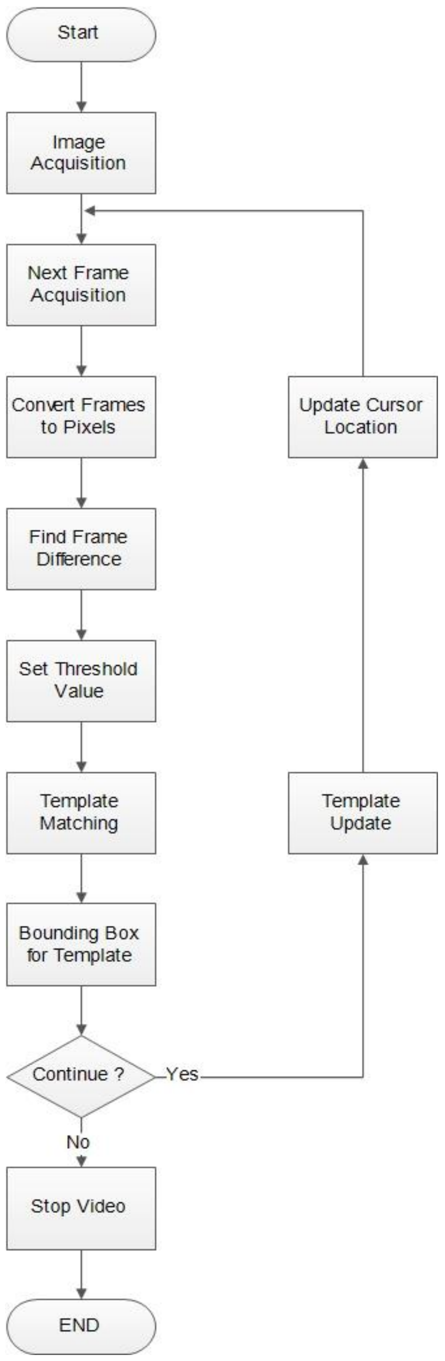
Fig. 2 The figure shows the neighbour window, the previous position of the motion template and matching region.

IV. Project Algorithm and Flowchart

A. *Algorithm*

step-0 : start
step-1 : image acquisition
step-2 : next frame acquisition
step-3 : convert frame to pixels
step-4 : find frame difference
step-5 : set threshold value
step-6 : template matching
step-7 : bounding box for template
step-8 : if don't want to continue go to step-12
step-9 : template update
step-10: update cursor location
step-11 : got to step-2
step-12 : stop video
step-13 : end

B. *Flowchart*



C. Details of the Algorithm

In order to implement this algorithm, the software tools selected are from the MATLAB Image Processing Toolbox (IPT), which occupies a position of eminence in both educational and industrial applications.

1) Image Acquisition:

- First the algorithm clears all variables and functions from memory and closes all open figure windows and clears the command window. All this is done to ensure robustness of system.
- Then video input object is created and then the video input object trigger settings are configured. Then it starts the timer running for video acquisition. Then it is paused for 2 seconds as in the first few seconds, web camera gives black frames as it needs some time to implement the settings configured.

2) Extraction of template for matching:

- After the pervious step is done the user is requested to choose any one way to specify the template.
- When the template is chosen, it is stored as the original template, otherwise continue to show the original frames
- If the template is provided it is marked with a minimum bounding box and a neighborhood box.

3) Matching of templates in subsequent image frames

- A threshold value is set to distinguish clearly the fairly matching template from the poorly matching ones.
- Next, neighborhood window for the template is created , taking into account the fact that the webcam is around 10 fps, so the movement of the feature (if any) would be minimum and mostly within the neighborhood box.
- Now comparison is done between the template and the extracted templates from the neighborhood box and finds the best match.
- The template is then updated as the new best match image.
- If the rate of best match is less than the threshold value, the template is saved and a search is done in some subsequent frames for a better match. If no match is found, then this frame is discarded and only the new frame is considered.

4) Controlling the cursor

- After getting the current position of the template, it is transformed to screen position using suitable transformations and control the mouse pointer accordingly.
- Then the algorithm aims to show the current frame with the traced template, and mark it accordingly. The loop is continued until the user clicked "stop". This way the current frame becomes the old frame for the next iteration of the tracking loop.

When the user clicks on the feature to be tracked, a square is drawn around the feature and the sub image within this square is cropped out of the image frame. The cropped sub image is used as a "template" to determine the position of the feature in the next image frame. To find this position, the tracking algorithm uses the template to search for the feature in a "neighbourhood window" that is cantered at the position of the feature in the previous frame. The template is shifted through this search window and correlated with the underlying sub images. The window is defined to contain the centres of all sub images tested.

Each sub image in the search window is matched with the template sub image that is cropped from the previous frame. As a measure of match, the normalized correlation coefficient

$$r(s,t) = \frac{A\sum s(x,y)t(x,y) - \sum s(x,y)\sum t(x,y)}{\sigma_s \sigma_t}$$

Where, $\sigma_s = \sqrt{A\sum s(x,y)^2 - \left(\sum s(x,y)\right)^2}$ and

$$\sigma_t = \sqrt{A\sum t(x,y)^2 - \left(\sum t(x,y)\right)^2}$$

and A is the number of pixels in template.

To quantify tracking performance, a match between a template and the best-matching subimage within the search window is called sufficient if the normalized correlation coefficient is at least 0.8. Correlation coefficients below 0.8 describe insufficient matches. Insufficient matches occur when the feature cannot be found in the search window because the user moved quickly or moved out of the camera's field of view.

V. Conclusion

A. *Uses*

It can be used in various places like:

- It can provide ways for computer human interface.
- It can primarily be used by physically disabled people to automate their interface.
- It can be used in now-a-days evolving smart classrooms for better interface. Since it uses a limited no of resources it can run at par with various soft wares under different hardware constraints.

B. *Screenshots*

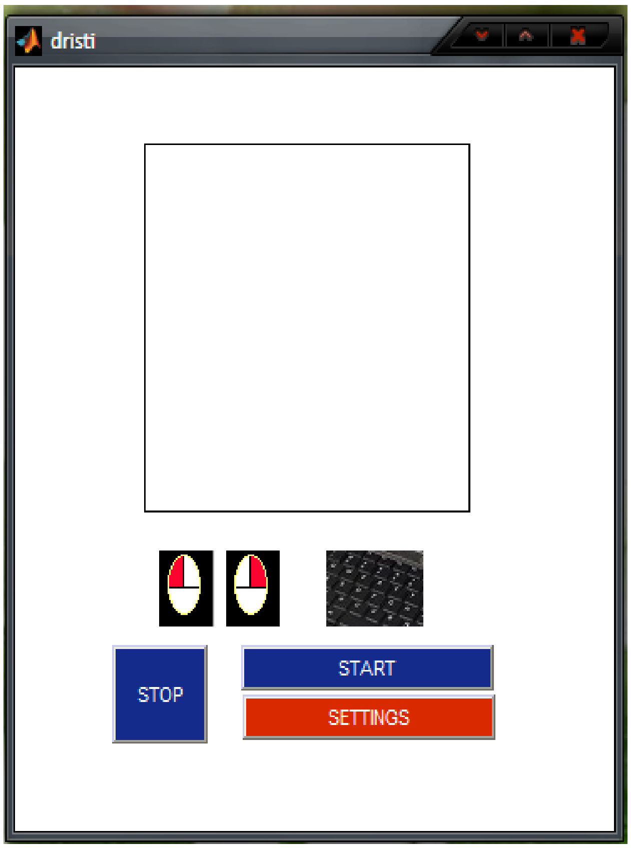
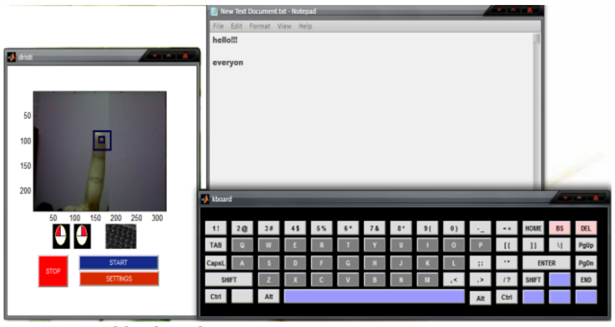


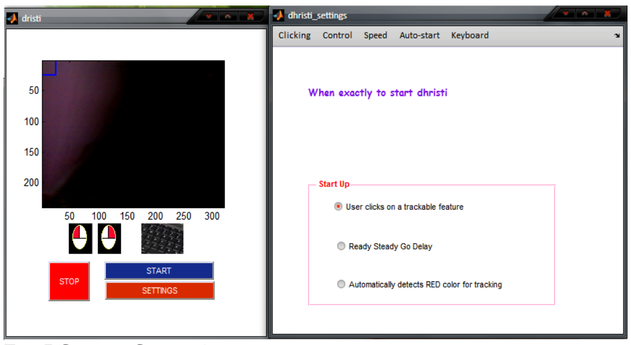
Fig. 3 The main screen of drishti. It contains the START STOP and SETTINGS options



hello!!

everyon

Fig. 4 Virtual keyboard

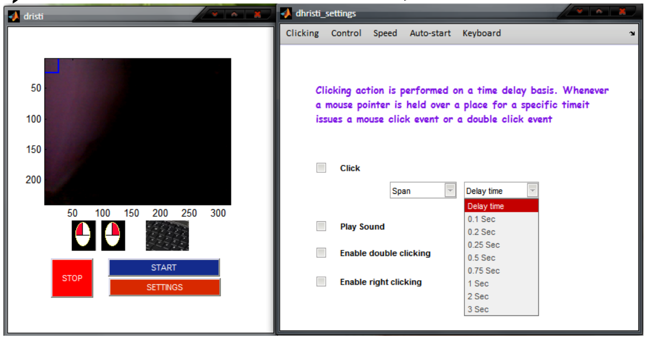


When exactly to start dhristi

Start Up

- [✓] User clicks on a trackable feature
- Ready Steady Go Delay
- Automatically detects RED color for tracking

Fig. 5 Settings Screen-1



Clicking action is performed on a time delay basis. Whenever a mouse pointer is held over a place for a specific timeit issues a mouse click event or a double click event

Click

Span

Delay time: Delay time, 0.1 Sec, 0.2 Sec, 0.25 Sec, 0.5 Sec, 0.75 Sec, 1 Sec, 2 Sec, 3 Sec

Play Sound

Enable double clicking

Enable right clicking

Fig. 6 Settings Screen-2

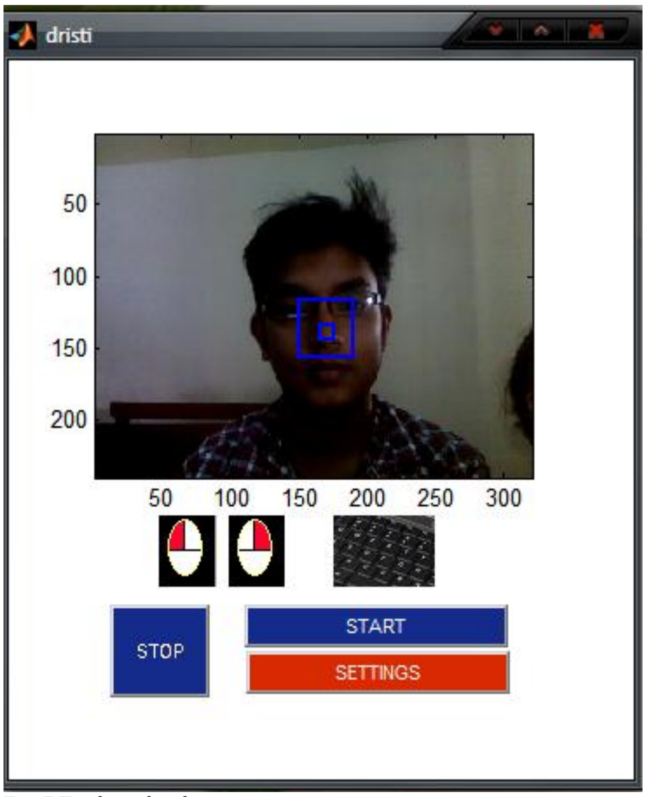


Fig. 7 Tracking the object

C. *Future Scope:*

To trace the eye movement of the user the algorithm can be modifies to trace eye gaze and that can be again used for a very efficient interface and can also play a vital role in intelligent systems. Secondly, the efficiency is moderate, but, if the algorithm is optimized a bit more, it might lead to a more efficient system design. Some more features can be incorporated to this software to make it functionally richer.

VI. References

[1]. Ankit Kumar, Ashish Joshi, Anil Kumar3, Ankush,Mittal4 and D R Gangodkar5" Template Matching Application In Geo-Referencing Of Remote Sensing Temporal Image", International Journal of Signal Processing, Image Processing and Pattern Recognition Vol.7, No.2 (2014), pp.201-210

[2]. Rafael C. Gonzalez "Digital Image Processing using MATLAB", Tata McGraw-Hill Education Private Limited, India, pp.74-78, 2010, ISBN: 13:978-0-07-070262-2.