RESEARCH ARTICLE OPEN ACCESS

Ary Mouse for Image Processing

[1]Vidisha Tembhurkar, [2] Darshana Paunikar [3] Akansha Rangari
[4] Shubham Ambade [5] Prof. Aniket Deo
Department of Computer Science and Information Technology, Nagpur Institute of Technology, Nagpur

ABSTRACT
In the field of touchy processing, many people would access the touchy phones, keypads etc. where the disadvantage of touchy system is all about touch screen. So, to overcome this problem, we are going to develop a project based on touch less device which is used to access and process our data with minimum time complexity for optimization of the sourcing data. Our project contain marker to highlight required key term. Touch less detects both the size and location of "Marker's Gestures" for writing purposes. Whereas the camera played a key role to select our object that is an image for it's processing. In this, we are giving command on camera to identify the gestures by Touch less SDK and reducing manual efforts.

Keywords: Touch less SDK, Neural Network, Handwriting Reorganization, Object Tracking, Hand as well as Marker detection, Image Processing.

I. INTRODUCTION

Touchless is an SDK that allows users to create and experience multi-touch applications. Touchless started as Mike Wasserman's college project at Columbia University. The main idea: to offer users a new and cheap way of experiencing multi-touch capabilities, without the need of expensive hardware or software. All the user needs is a camera, which will track colored markers defined by the user. Mike presented the project at the Microsoft Office Labs Productivity Science Fair, Office Labs fell in love with it, and Touchless was chosen as a Community Project. Our deliverables include an extensible demo application to showcase a limited set of multi-touch capabilities, but mainly we are delivering an SDK to allow users to build their own multi-touch applications. Now, Touchless is released free and open-source to the world under the Microsoft Public License (Ms-PL) on CodePlex. Our goals are to drive community involvement and use of the SDK as it continues to develop. Remember that this is just the beginning; and you're invited to join our journey. Send us your questions and feedback, use Touchless SDK in your .NET applications and XNA games, and support the community by contributing to the source code.

II. METHODOLOGY

This release includes a short list of binary files to demonstrate Touch less:

- WebCamLib.dll - Interfaces with DirectShow to grab webcam frames
- TouchlessLib.dll- Contains the functionality of Touch less SDK
- Image.gif - Used in Touch less Demo's image demo
- Touchless.chm - A documentation file of the Touchless API for developers
- ouchless.jpg - A class diagram of the Touchless API for developers

III. HANDWRITING ECOGNIZATION

Handwriting recognition well fits into this category. Handwriting recognition is the task of transforming a language represented in its spatial form of graphical marks into its symbolic representation. On the one hand, it is convenient and useful to write using a pen, track its movement through a camera, and transform the writing into digital document. On the other hand, Touchless works well for handwriting tracking because inaccuracy does not count that much in this– human handwriting, as well as many other human activities, is fundamentally not accurate. (Fig. 1) Not only do different people have different types of handwritings, even if for the same person, the script could be different in different time or when he/she is in different mood. Compared to the inaccuracy introduced by the above factors in handwriting, the inaccuracy of Touchless is not considerably significant. Handwriting data is converted to digital form either by scanning the writing on paper or by writing with a special pen on an electronic surface such as a digitizer combined with a liquid crystal display. The two approaches are distinguished as off-line and on-line handwriting, respectively. In our case, we choose on-line recognition in this case because Touchless tracks position data of object in real time and on-line recognition uses two-dimensional coordinates of successive points of the writing as a function of time. The data can then be processed by pre-trained Artificial Neural Network and recognized.

Fig. 1. Handwriting Samples

IV. IMAGE MOVEMENT

Create a utility function to retrieve Image Data in a consistent manner; we have a bit of code duplication right now. Make a public interface for demo classes to implement, then allow the user to just invoke start and stop of a class on the library Standardize error handling and exception generation across the project. Improve the expected marker regions used for scanning on update. We could consider the marker's acceleration, rather than just the velocity. Perhaps try using regions that aren't axis-aligned rectangles.

V. HARDWARE AND SOFTWARE DETAILS

5.1 C#.NET

C#.NET (**C#.NET**) is an object-oriented computer language that can be viewed as an evolution of Microsoft's implemented on the Microsoft .NET framework. Its introduction has been controversial, as significant changes were made that broke backward compatibility with C# and caused a rift within the developer community.
The great majority of C#.NET developers use Visual Studio .NET as their integrated development environment (IDE). Programs written in VB.NET require the .NET framework to execute.

5.2 HARDWARE

Laptop
Mouse
Keyboard
Web camera

VI. DESIGN DETAILS

By adding Touchless to your project, you can give your users a truly fun, novel, and functional multi-touch experience, and all they need is a webcam and any computer camera.
The prerequisites you'll need to develop using Touchless are:

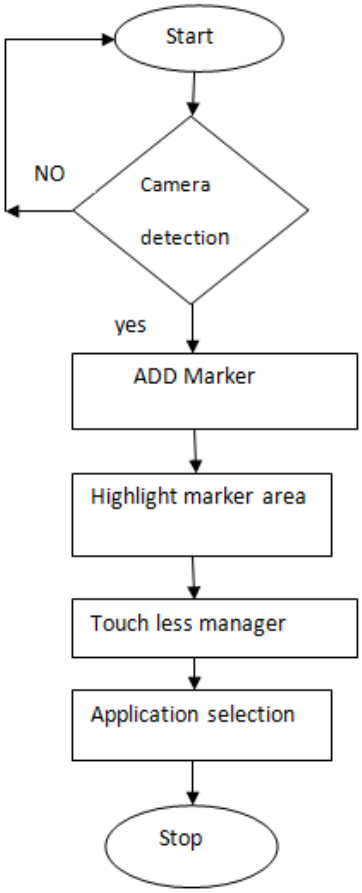
- Visual Studio 2010 or 2012
- .NET 3.0 or higher
- "TouchlessLib.dll" and "WebcamLib.dll"

Following are the process to set-up the "TouchlessLib.dll" and "WebcamLib.dll" from an .NET framework.

To add Touchless to an existing visual Studio project, simply right click "References" and select "Add Reference…" go to the browse tab, and select "TouchlessLib.dll".

Ensure that both "TouchlessLib.dll" and "WebcamLib.dll" are copied to the same output directory to be used with your builds.

VII. FLOW DIAGRAM



7.1 Implementation

7.1.1. Color Lib

Improve HSV color space partitioning model. We could group perceived similar colors better. Potentially replace with a group clustering algorithm. Perhaps just refine the per-dimension bin counts, or replace the hash function.

Use a lookup table instead of transforming RGB to HSV. We can just terminate early if it's not in the lookup table.

Reduce loop overhead of converting ARGB values into RGB values, then into HSV values, then into Binned HSV values, then finally into a hash # for color lookup during marker update. Potentially use a lookup table for a subset of colors to avoid the math altogether.

Improve HSV color grouping, consider refining the per-dimension bin counts or using a different HSV color-space partitioning model that better suits human perception of similar colors.

7.1.2. Marker

Implement a way of getting higher degree moments of inertia. Mostly, we are interested in the axis of least rotational momentum and the roundness factor.

Allow the user to send a mask image with the add marker bitmap for arbitrary marker region selections."

Extend or replace alpha smoothing with exponential decay to provide smoothed marker data and reduce the marker jumpiness.

Optimize threshold, or replace threshold concept with a partial matching. Also, step threshold by numbers that actually make a difference, or just have sensitivity +/- buttons and increment functions Expose smoothing factor as a public marker property.

Fix and improve the automated marker tests Standardize some marker colors, create an "auto-find makers" Improve the search bounds of a previously absent marker.

Improve the meta-tracking (cases where small numbers of pixels are missing from the middle of a marker, or are outliers of the concentration of pixels) Periodically/continuously adopt surrounding pixels of confirmed marker pixels Coloravg is currently just marker representative color. Implement a way of actually getting a color average from the set of colors found Improve Marker highlighting Improve upon the raster scan algorithm used for marker updating.

Optimize the method for getting the marker appearance from a circular area of a bitmap; we could use hierarchical bounds intersection or something smarter than the current scan algorithm.
Optimize the values used to increment/decrement color frequencies for marker appearance detection. This should be somehow based on signal/noise ratios.

Improve the expected marker regions used for scanning on update. We could consider the marker's acceleration, rather than just the velocity. Perhaps try using regions that aren't axis-aligned rectangles.

7.1.3. Touchless Manager

Add functionality to save and load marker configuration files (reduce repeat training of the same marker, possibly provide autoconfig files for standard markers... will variant lighting allow for this?) Implement additional marker data such as ColorAverage, ColorSpace, Axis and Roundness.
Add flood fill algorithm so we can add a marker with a few points in the Bitmap.
Refine the marker tracked colors as we find colors around the marker.
The representative color doesn't always match the perceived color of the marker.
Provide subsequent examples of a marker appearance.

Have TouchlessMgr actually expose a way to get a list of the current markers Make a better exception for camera start failure Validate the PixelFormat of incoming images.

Create a utility function to retrieve ImageData in a consistent manner; we have a bit of code duplication right now.

Make a public interface for demo classes to implement, then allow the user to just invoke start and stop of a demo class on the library.
Standardize error handling and exception generation across the project.

VIII. CONCLUSION

In the designing of our projects, we have kept in mind the user in the implementation part which interacting with the user we had given lot of guideline to user with various massages. Net very good programming languages for implementation of any data base projects because it has powerful control with which you can easily implement various facilities in our projects .the screen are very user friendly.

IX. FUTURE SCOPE

Technology is called **"touchless, telepresence display"** and was performed by placing a video camera under a transparent OLED display. The screen, measuring less than 2 inches thick and has a 40-inch diagonal, and is thus able to display and simultaneously detect gestures made above the surface or direct touches on the surface. Camera sees through the screen and is thus practically able to register user's presence and gestures. According to researchers from **Applied Science Group at Microsoft**, this technology will change the displays passive role. If now they serve more to show, in the future will become true **"windows"** through which the user can interact with the digital world

REFERENCES

[1]. NET by Evangelos Petroutsos
[2]. NET Programming by vikas gupta.
[3]. www.microsoftlabs.com.
[4]. Niesink, L. Adding A Third Dimension To Multi-T Table Top Display Interaction, 2010.
[5]. Ryu, D. , Um, D, Tanofsky, P. , Koh, D. H., Ryu, Y. S.,& Kang, S. (2010). T-less: A novel touch less Human.
[6]. Machine Interface based on Infrared Proximity Sensing In Proceedings of 2010 IEEE/RSJ International Conference on Intelligent Robots and Systems (pp.5225225)Taipei, Taiwan.
[7]. Pickering, Carl A. Burnham, Keith J.

Richardson Michael J. Jaguar ,"A research Study of Hand Gesture Recognition Technologies and Applications for Human Vehicle Interaction", 3rd Conference on Automotive Electronics, 2007