Comp 437 Intelligent User Interface Term project. Smart Pen.

Ahmed Gamal Masry

Koc University.

Abstract.

This report describes the work done for Comp 437 Intelligent user interface term project. Smart Pen is a project that enables the user to use a normal pen and a blank A4 paper instead of a stylish pen to capture hand-written texts and drawings using the webcam of his/her laptop as the input device. It alleviates the burden of buying the expensive stylish pen and the board.

1. Introduction.

Since the rise of the Digital era, technological devices have become an essential part of our daily life. Every day we use many smart electronic devices to facilitate our work. During our interaction with the electronic devices, we need to use input equipment to deliver our commands to the devices. A novel input peripheral that has been introduced to the market recently is the Stylus-pen device. In computing, a **stylus** (or **stylus pen**) is a small pen-shaped instrument that is used to input commands to a computer screen, mobile device or graphics tablet [1]. Here we tried to develop a smart software system that enables us to use a normal pen with a blank A4 paper instead of the stylish pen to achieve the same purpose by tracking the pen header movements. In the following section, we describe our system architecture and how it works. Afterwards, we present our results.

2. Technical Aspects.

Smart Pen uses the computer vision techniques implemented in the OpenCV library in python to detect and track the pen header to recognize what the user is writing or drawing.

Figure 1 describes the flow of our algorithm. At the beginning, the user should make sure that all the paper edges are inside the view of the webcam as displayed on the screen. In step 1, the user should click the "Detect Paper" button to search for the paper in the current scene captured by the camera. Firstly, we convert the frames from RGB to grayscale, and detect the edges in the grayscale image. Secondly, we find the contours between the edges in the image. Lastly, we iterate over them to find a rectangle-shaped contour with only four corners that are at least 200 pixels apart. We then save these four points since they define the corners of our paper. In step 2, the user should click the "Pen header" button to locate the pen header location on the paper. Firstly, we convert the frames from BRG to HSV format. Secondly, we use the four corner points found above to transform the paper found in the frame onto the full screen. Thirdly, we search for the pen header color (red) by checking the pixels that have colors in a specific range of hsv values. Once we find pen header location, we save it so that we can track it later. In step 3, the user should click the "Start drawing" button to track the pen header using the Lucas-Kanade optical flow algorithm.

In computer vision, the Lucas–Kanade method is a widely used differential method for optical flow estimation developed by Bruce D. Lucas and Takeo Kanade. It assumes that the flow is essentially constant in a local neighborhood of the pixel under consideration

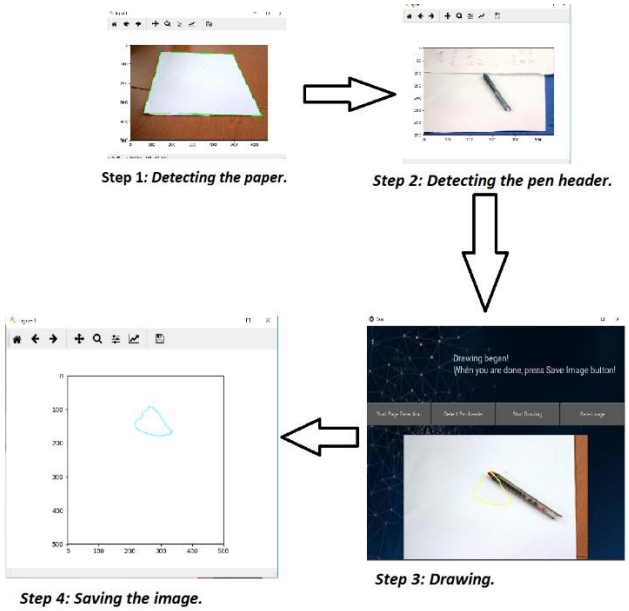

Figure 1: Flow of the system.

and solves the basic optical flow equations for all the pixels in that neighborhood, by the least squares' criterion [2].

The algorithm was given the pen header location at the beginning, and it tracks the next points for that frame using the parameters specified in the code. The new points of the pen header are all stored in one list, so that they can be used to draw lines connecting them to represent the user drawings. In step 4, the user clicks the "Save Image" button to save the drawings in a jpg image in the same directory of the project.

Unresolved challenges.

However, the system couldn't achieve the required task efficiently for several reasons. Firstly, the number of frames per seconds the camera could record was only 30 fps. When the user moves the pen so fast (which is the case for during writing), the camera gets distorted pictures in which the Lucas-Kanade algorithm can't track the pen header easily since it's a violation of its assumption. Secondly, the system has no way of detecting whether the pen is in contact with the paper or not during drawing. We have tried to fix this issue with two solutions.

The first one is to use dejavu library to memorize the sound fingerprints of the pen contact with the paper in a database. Then when getting the records from the microphone, we try to match them with the audios stored in the database. However, the performance wasn't good enough; the library couldn't distinguish between silence and the sound of the pen contact with the paper. Hence, this method failed to fix our problem.

The second one is to extract some features from the recorded audios and apply a machine learning classification algorithm on them. We used pyaudioanalysis library to extract the features from the recorded audios. However, when we visualized the features using Pandas, we found out they aren't reliable at all to classify silence from the contact sound.

Criteria	Average
The interface was simple and easy to use	4.4
The method of writing and drawing seems natural.	2.6
Recovering from errors was easy.	3.6
Useful notification were provided when needed.	4.2
The flow of the process was smooth.	3.8
The limited speed of drawing or writing was enough for the interface to be natural.	2.6
The method of detecting the pen header was convenient.	3.0
The program enabled you to achieve the task efficiently.	2.6
How was the overall experience with the interface?	3.4

	Value
Extremely Poor	1
Poor	2
Neither good nor poor.	3
Good	4
Very Good	5

Figure 2: Evaluation results

3. Novelties.

The system could have distinguished itself by integrating some hardware (some nuts for example) on the pen in addition to the intelligent software. Such hardware can enable the user to control some properties of the text like the font size, color… etc. However, most of the time was invested on building the software part, and we didn't have the chance to come to the novelties part.

4. Technologies Used.

The project was mainly implemented in python 3. The user interface has been designed using Kivy API. We have also used the pyimagesearch software package to perform some image processing tasks such as transforming images. Numpy, and the famous matplotlib were also used. However, most of the work has been done using OpenCV library. Although our system works using OpenCV methods, deep learning methods would have been much better to achieve this task.

For the hardware, we have only used a webcam for recording the video, a cheap pen with a red header, and a blank A4 paper.

5. Results and Evaluation.

As shown in figure 1, the system could track the user handwriting except that it doesn't detect when the user lifts his pen from the paper because we couldn't implement a reliable algorithm to get the distance between objects in 3D space from a 2D frame. We believe the project can be completed successfully using Kinect sensor since it gives information about the depth. We conducted a survey with 5 students who evaluated our IUI system. We briefly explained the purpose of our system and how to use it to them. Then they were asked to use and evaluate it. The evaluation metrics, their values, and meanings are shown in figure 2.

6. Related Works.

Paper detection in images has been an interesting problem for computer vision engineers due to its benefit in many applications. Thus, many approaches were proposed to solve this problem. One possible approach was proposed by Maephisto [3] in which he applies gaussian blur and canny, find the contours and pick the biggest most rectangle shape in contour. This approach was adopted in our system with some small modifications so that it fits our problem.

Tracking objects in the consecutive frames has been a very interesting problem as well. A very efficient method that was introduced was Lucas-Kanade optical flow algorithm. It has been implemented in opencv and its usage became mush easier. In the official tutorials of opencv, a very simple example was given [4]. This example has been used in our code after some small changes to fit the purpose of our system.

7. Acknowledgements.

At the end, I would like to thank Prof. Metin Sezgin and all my classmates for their feedback during all the presentations. Although the project idea couldn't be implemented successfully, I believe that after acquiring more advanced skills in Computer vision, Digital audio processing, and deep learning, I will be able to achieve it with a higher quality.

8. References.

[1] Wikipedia contributors. (2019, March 30). Stylus (computing). In *Wikipedia, The Free Encyclopedia*. Retrieved 05:35, May 30, 2019, from https://en.wikipedia.org/w/index.php?title=Stylus_(computing)&oldid=890143634

[2] Wikipedia contributors. (2018, June 8). Lucas–Kanade method. In *Wikipedia, The Free Encyclopedia*. Retrieved 05:36, May 30, 2019, from https://en.wikipedia.org/w/index.php?title=Lucas%E2%80%93Kanade_method&oldid=845039608

[3] Maephisto. (2014, September 14). Extracting A4 sheet out of troubling backgrounds. Retrieved May 11 2019, from https://answers.opencv.org/question/174235/extracting-a4-sheet-out-of-troubling-backgrounds/

[4] OpenCV. Optical flow.

Retrieved May 16 2019, from https://docs.opencv.org/3.4/d4/dee/tutorial_optical_flow.html