ISSN 2278-3091
Volume 8, No.4, July – August 2019
International Journal of Advanced Trends in Computer Science and Engineering
Available Online at http://www.warse.org/IJATCSE/static/pdf/file/ijatcse29842019.pdf
https://doi.org/10.30534/ijatcse/2019/29842019

GPU Accelerated Stereo Vision System to Capture Handwriting Information

Fadi Imad [1], Sharifah Mumtazah Syed Ahmad [2], Shaiful Jahari Hashim [3], Khairulmizam Samsudin[4]
Faculty of Engineering, Universiti Putra Malaysia, Serdang, Selangor, Malaysia
[1]fadiaburaid@gmail.com
[2]s_mumtazah@upm.edu.my
[3]sjh@upm.edu.my
[4]khairulmizam@upm.edu.my

ABSTRACT

Stereo vision systems that capture 3D space data are of great use as computer interface systems for emerging technologies such as VR and Robotics. They are inexpensive, yet practical implementation is challenged by high CPU processing power requirement. Thus, in this paper a scheme is developed to exploit GPUs in the processing of high definition images required to track an object position and orientation in 3D space in real-time. This is done by porting and optimizing essential processing algorithms to operate efficiently in parallel on a GPU. The object detected is a pen, with the position of the tip and it's orientation tracked in real-time. The writing captured by the system is than compared to the one from commercial digitizing tablet. The results achieved indicated that pen tip can be tracked in 3D space with a percentage error below 1.7% within a distance of 40cm from the stereo camera shooting at a frame rate of 30 frames per second.

Key words : Stereo vision, real-time vision, parallel computing, 3D tracking.

1. INTRODUCTION

Traditional human computer interface devices such as knobs, sliders, joysticks, keyboards and mice have been for long the most commonly used for many tasks. However, these interface devices fail to meet the requirements or cripple many emerging technologies which require manipulation in 3D space such as virtual reality and control of robots and drones etc. Many researchers proposed visual techniques as a viable solution for 3D tracking and robotics control as these require no special hardware beside the widely available consumer grade web cameras. A similar system is proposed where cameras are used to construct a low-cost stereo vision system to track the position and orientation of an object in 3D space. To ease the process of testing the system performance the object to be tracked was chosen to be a pen so the actual pen traces could be compared against captured motion of pen tip.

Stereo vision systems which capture handwriting has been previously investigated such as systems by M. Moriya et al.[1], Dave Jay et al.[2]and Fadi Imad et al..[3] Yet these systems were challenged by the inability of CPU to process stereo HD frames at a high frame rate for accurate real-time tracking. This can be visible in the data points being sparse thus the generated writing becomes unclear. Among the solutions implemented to solve this matter was resampling the data to compensate for the drop of frame rate like in the work done by G. A. Fink et al. [4]. Yet, this solution was only viable when the system is not operating in real-time. Since real time stereo vision requires a lot of processing power to achieve high accuracy and high frame rate simultaneously. Thus, the use of GPU is proposed to allow for higher accuracy and frame rate.

The structure of GPU consists of thousands of less powerful cores, unlike CPU which has a few powerful cores. These cores allow the GPU to offer more processing power by allowing many sessions of the program to run in parallel as shown in Figure 1.

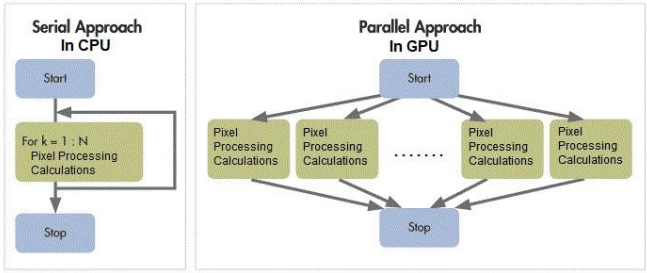


Figure 1: GPU Processing Versus CPU

This is very efficient when the same computation has to be performed on many elements of the data, which is the case of many image processing algorithms that operate on pixels independently or rely only on the neighborhood around pixels. Since many desktop computers and laptops nowadays come with fairly powerful Graphics Processing Units utilizing them will not inflict any additional cost. Researchers such as Hongjian Wang et al.[5] and Ge Li et al.[6] have utilized GPU in stereo vision, however their applications required a full depth map which requires huge amount of computation to solve correspondence problem which is not

the case with our application of stereo vision as computer interface. For tracking object's location and orientation in 3D space tracking 2 points is sufficient.

This paper presents an efficient parallel processing scheme to allow for the use of GPU in Pen tip position and orientation tracking. The developed structure enables visual pen input capture in real time. The paper is organized in the following order: In Section 2 the system components and their configuration are described. Then in Section 3 the algorithms applied are explained. After that, in Section 4 the system performance was evaluated. Finally, the findings and observations are presented in Section 5.

2. SYSTEM DESCRIPTION

2.1 System components

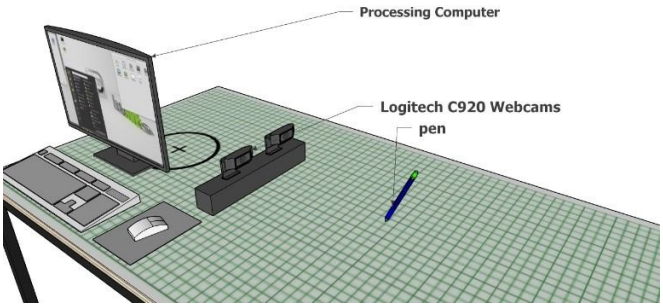


Figure 2: Experimental Setup

The system hardware consists of three major components as displayed in Figure 2. The main component is the stereo camera pair which is made of two identical of the-shelf webcams. The webcams have a resolution of up to 1920×1080 at 30 frames per second. They were mounted as close as possible to the coplanar configuration with a lateral displacement of 10 cm.

Figure 3: Stereo Camera Configuration

The second component is the pen to be tracked. To allow for detecting the pen position based on color filtering the pen was labeled with two colored markers made of non-glossy colored tape to avoid ambient light reflection which affects the color contrast. Green and yellow colors were purposely chosen for the markers in order to avoid interference from background objects as green and yellow objects are rarely used indoors. [7] Connected Component Labeling (CCL) algorithm which is used in detecting blobs is serial by nature as it performs two raster scans when processing pixels. To avoid this, a parallelized CCL algorithm by Ondˇrej ˇS´tava et al. [8] was tested, and the improvement in comparison to using CPU was found to be minor. The algorithm reduced processing time by slicing high resolution images into smaller tiles which are then processed by GPU threads. However, this step was followed by a slower iterative merging process to give the connected components in different tiles the same labels by processing the boundaries of tiles. Therefore, CCL was abandoned and two labels of different colors on the pen were used to differentiate the two blobs.

The final component is a desktop computer to process the captured images. The computer is equipped with a quad core i7 processor, 8 GB of RAM and Nvidia GTX 1080 graphics card with 2560 CUDA cores. Image processing and digitizing software was developed in C++ using OpenCV library to capture and display images.[9] The actual image processing algorithm kernel was written in CUDA C. The system components were placed on a desk which also represents the writing surface. The desk is large and covered with grid paper to ease the testing and evaluation of the detection range and accuracy.

2.2 Stereo camera system calibration

The first crucial step that precedes the use of stereo cameras is the calibration process. Calibration was performed to compute the extrinsic and intrinsic parameters of the camera. The extrinsic parameters of a camera specify the position and the orientation of the camera pair with respect to each other in the coordinate system. The rotation matrix generated enables us to rotate and rectify the image frames to be coplanar although the cameras are not in perfect coplanar configuration. In addition, the translation matrix allows us to find the baseline displacement between the cameras to higher accuracy. The intrinsic parameters describe the inherent properties of the camera optics, including the focal length, the image center, the image scaling factor and the lens distortion coefficients. Distortion coefficients are used to remove the distortion due to lens imperfections which produces curved lines where straight lines should take place. Whereas the other parameters are used to solve for 3D information in the later steps. The intrinsic and extrinsic parameters can be found by relating the 3D points in the real world and their corresponding 3D projections. For this an object of known geometry had to be used. The calibration methods we followed used the rectangular checkerboard patterns (Tsai grid[10]) as shown in Figure 4, while depending on Zhang's [11] Camera Calibration algorithm to solve the following equation:

$$C_n = K[R|T]W_n \quad (1)$$

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} a_x & 0 & u_0 \\ 0 & a_y & v_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_1 \\ r_{21} & r_{22} & r_{23} & t_2 \\ r_{31} & r_{32} & r_{33} & t_3 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \quad (2)$$

u and v represent the point projection in pixels, X, Y and Z are known as world homogeneous coordinates.[12] While K and $[R|T]$ represent the intrinsic and extrinsic parameters respectively. The extrinsic parameters matrix consists or R which is the rotation matrix and T which is the translation matrix. (u_0, v_0) is a principal point which represent the center of the image and a_x, a_y are focal lengths expressed in pixel-related units. r_{nn} are the rotation parameters, while t_n represents the translation parameters.

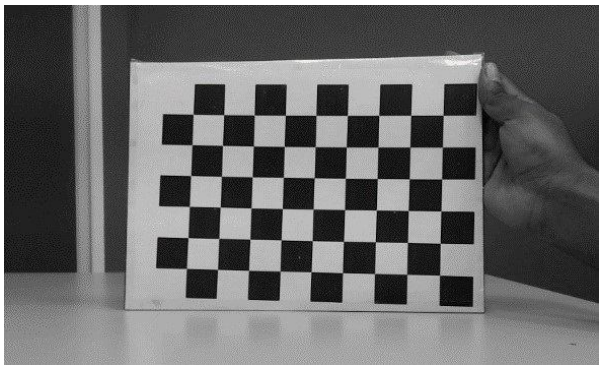


Figure 4: Stereo Camera Calibration Checkerboard

During the calibration process multiple images of the calibration checkerboard were captured in different orientations. The edges of the black squares were detected and used along with the known size of the square in real world to solve for the calibration parameters. The parameters were used to construct the 3D projection matrix which was then saved in a file for later use.

3. TRACKING ALGORITHM

The algorithm developed tracks pen tip position and orientation in the 3D space from a pair of 2D images. The images used for tracking were captured at a resolution of 1280×720 pixels. Each camera captures 30 video frames per second. The sequence of processes followed in the algorithm is displayed in Figure 5. Initially the camera calibration matrices were read from the files generated earlier and were used to construct transformation maps. Then a program loop was initiated, and frames were captured. After that the frames were undistorted and rectified using the maps constructed previously. Next the colored labels were filtered from the rest of the image using Hue, Saturation and Value (HSV) color space. Then in steps 6 to 7 morphological operations were performed in filtering noise and in edge detection. Following that the centroids of the detected labels were found. After that, the centroids and edges were used to detect the pen tip in each frame. Finally, the pen tip positions in the two frames were used to find the 3D position and orientation of the tip using triangulation.

To achieve 3D tracking while maintaining a high frame rate the GPU was used when repetitive calculations were performed on image pixels. This was the case with steps 4 to 9 of the algorithm. On the other hand, the CPU was used when a single calculation was performed to utilize the higher processing speed and to avoid the unnecessary latency associated with copying variables to GPU memory. Furthermore, all steps which does not have to be repeated in the capturing and processing loop such as variable declarations and generation of transformation maps were placed outside the loop.

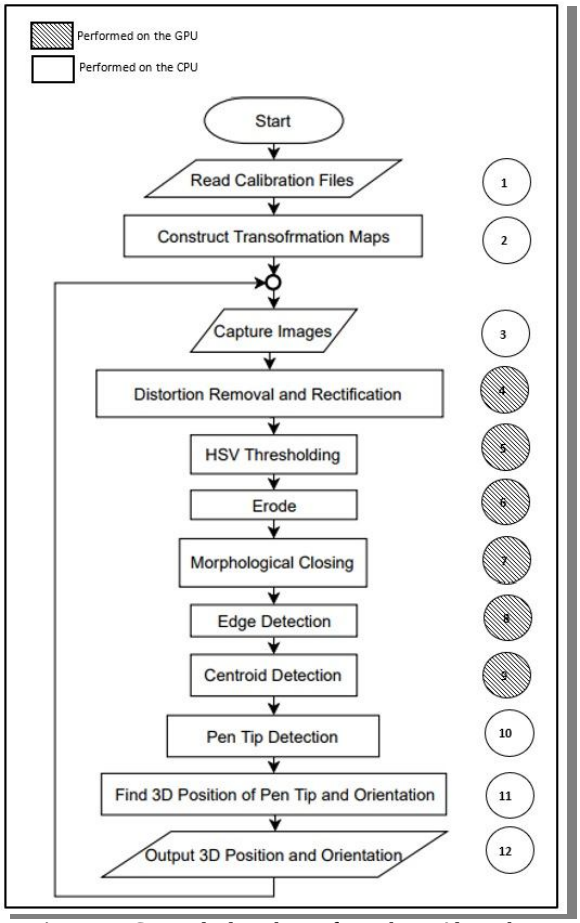


Figure 5: General Flowchart of Tracking Algorithm

3.1 Generating transformation maps

The calibration files generated earlier which contain the calibration parameters matrices were read into 2D arrays. To avoid unnecessary repetition of complex matrix calculations in the GPU whenever the corrected location of a pixel is to be found after distortion removal and rectification, transformation maps which contain the position of the pixel pertain to the corrected pixel in the original image. This was done by firstly removing the distortion. There are two types of distortion considered radial and tangential. Radial distortion can either be negative (pincushion) or positive (barrel) and it increases the further the pixel is from the image center as shown in Figure 6.

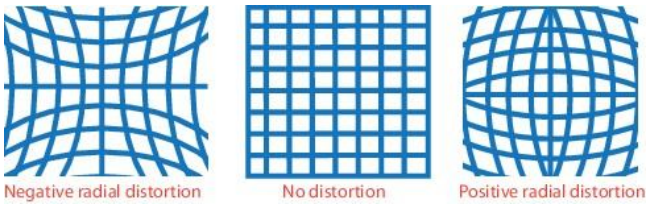


Figure 6: Types of Radial Distortion

Whereas tangential distortion is caused by the lens not being perfectly aligned with the camera sensor. Using the distortion coefficients, the distortion could be corrected according to the following equations:

For radial distortion

$$x_{corrected} = x(1 + k_1 r^2 + k_2 r^4 + k_3 r^6) \quad (3)$$

$$y_{corrected} = y(1 + k_1 r^2 + k_2 r^4 + k_3 r^6) \quad (4)$$

For Tangential distortion

$$x_{corrected} = x + [2p_1 xy + p_2(r^2 + 2x^2)] \quad (5)$$

$$y_{corrected} = y + [p_1(r^2 + 2y^2) + 2p_2 xy] \quad (6)$$

Where (x, y) are the coordinates in the input image and ($x_{corrected}$, $y_{corrected}$) is the position on the corrected output image and r is the radial distance from the center of the image. k_1, k_2 and k_3 are the radial distortion coefficients, while p_1 and p_2 are the tangential distortion coefficients. These coefficients were obtained through the calibration process (see Sect.2.2). After the undistortion was described in the transformation maps the rotation and translation matrices were used to make both camera image planes coplanar. This makes all the epipolar lines parallel and allows for fining depth using triangulation. The pixels are shifted to simulate rotating cameras vertically and horizontally to achieve zero angles between their 3D axes. Then translating a camera to have zero translation between their axes in all directions except the direction of baseline displacement. Finally, these translations and translations were incorporated into the transformation maps.

3.2. Undistorting and rectifying images

After the transformation maps were created the algorithm enters into the capture and image processing loop. Two images were captured by the stereo camera then passed to the GPU along with the x and y transformation maps of each to be undistorted and rectified. Each GPU thread takes a pixel from the target image find the location of its corresponding pixel in the original image from the x and y transformation maps. Then the pixel value is copied from the original image and placed in the target pixel as illustrated in Figure 7. The same process was repeated for all the target image pixels in parallel and since the process involves only coping of pixel values and no repetitive complex undistortion and transformation calculations this increases the speed considerably.

3.3 HSV thresholding

The next step after correcting for distortion and camera transformations was to detect the colored markers on the pen. To do so the image color representation is changed from RGB to HSV which is good in separating color components and more robust to lighting changes.[13] Then the colors of the two labels were filtered and labeled on a single channel grayscale image frame. Green pixels were given a value of 255 (white) and yellow were given the value of 127 (gray). This process was written in a CUDA C kernel to run once for every pixel in a separate GPU thread. Algorithm 1 presents the pseudo-code and the output is displayed in Figure 8.

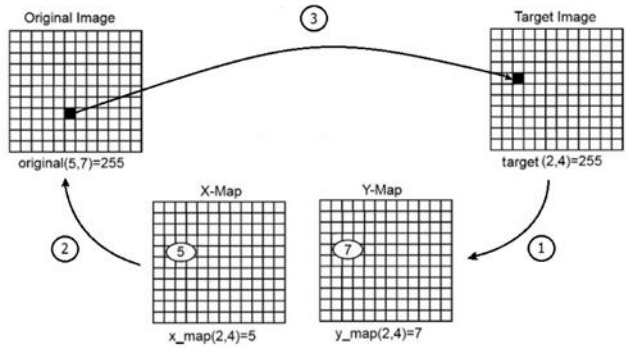


Figure 7: Image Undistortion and Rectification Process

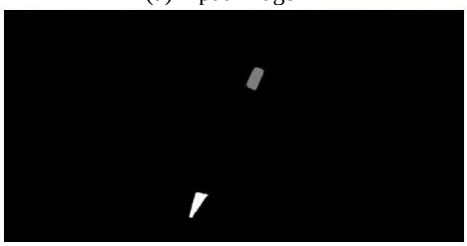
Algorithm 1:

```
r ← rgbaPixel[xcoordinate; ycoordinate; channel1]
g ← rgbaPixel[xcoordinate; ycoordinate; channel2]
b ← rgbaPixel[xcoordinate; ycoordinate; channel3]
maxrgb ← maximum(r, g, b)
minrgb ← minimum(r, g, b)
diff ← maxrgb - minrgb
V ← maximum(r, g, b)
if V = 0 then
   H ← 0
   S ← 0
else
   S ← diff/V
if diff < 0.001 then
   H ← 0
else
   if maxrgb = r then
      H ←60 * ((g * b)/diff)
   if H < 0 then
      H ← H + 360
   else if maxrgb = g then
      H ← 60 * (2 + ((b - r)/diff))
   else
      H ← 60 * (2 + ((b - r)/diff))
S ← S * 255
H ← H/2
if (H > 45 & H < 90) & (S > 100 & S < 255) & (V > 50
& V < 255) then
   hsvImage[x, y] ← 255
else if (H > 20 & H < 30) & (S > 100 & S < 255) & (V
> 50 & V < 255) then
   hsvImage[x, y] ← 127
else
   hsvImage[x, y] ←   0
```

Algorithm 2:

```
pxldilate ← dilateImage[x, y]
pxlerode  ← erodeImage[x, y]
if (pxldilate = 255) & (pxlerode = 0) then
   edgeImage[x, y] ←  255
```

(a) Input image



(b) Output image

Figure 8: HSV filtering Operation

3.4. Erosion and morphological closing

Morphological operations were applied to each pixel with either a value of 255 or 127 of the output from the previous step in parallel using a GPU kernel. Firstly, erosion was applied to remove any noise from the background. Then morphological closing is applied to fill the gaps in the connected component. This was done by applying a dilation operation followed by erosion. The operations were applied using the structuring elements in Figure 9.

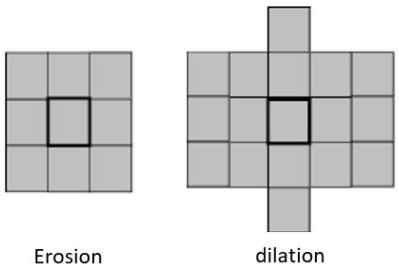


Figure 9: Structuring elements used

3.5 Edge detection

The edges of the labels were detected from the output of HSV filtering. Finding the boundaries of the blobs was done so it could be used in the final stages to locate the pen tip which lies on the boundary of the green label. The edge was found by finding the difference between corresponding pixels in eroded and dilated images as shown in Algorithm 2.

3.6 Find connected components centroid

The centroids were found from the detected edges because processing only edges pixels will contribute in reducing processing time. To find the centroid of the blob, it is defined as a point, whose x value is calculated by summing the x coordinates of all pixels in the blob and then dividing by the total number of pixels and similarly for y. In mathematical terms the centroid, (x_c, y_c) is calculated as follow:

$$x_c = \frac{1}{N}\sum_{i=1}^{N} x_i \tag{7}$$

$$y_c = \frac{1}{N}\sum_{i=1}^{N} y_i \tag{8}$$

The total number of white and gray pixels and the sum of x and y values of each were found using the GPU as shown in Algorithm 3, while the final calculation was performed on the CPU. This was done since the final calculations were not repetitive.

3.7 Detecting the pen tip

To finally detect pen tip in each frame, the equation of the line connecting the two centroids was found. Then the line was extended from the centroid near the tip of the pen until it crosses the edge. The position of the tip was at a small distance from the crossing point as shown in Figure 10. This process is iterative however due to the number of pixels along the line being small it can be performed on the CPU without impacting the performance.

Algorithm 3:

```
ngraypxl ← 0// number of gray pixels
sumgrayx ← 0// sum of x of gray pixels
sumgrayy ← 0//sum of y of gray pixels
nwhitepxl← 0// number of white pixels
sumwhitex← 0// sum of x-coordinates of white pixels
sumwhitey← 0// sum of y-coordinates of white pixels
if edgeImage[x; y] = 127 then
   ngraypxl← ngraypxl + 1
   sumgrayx← sumgrayx + x
   sumgrayy← sumgrayy + y
else if edgeImage[x; y] = 255 then
   nwhitepxl← nwhitepxl + 1
   sumwhitex← sumwhitex + x
   sumwhitey← sumwhitey + y
```

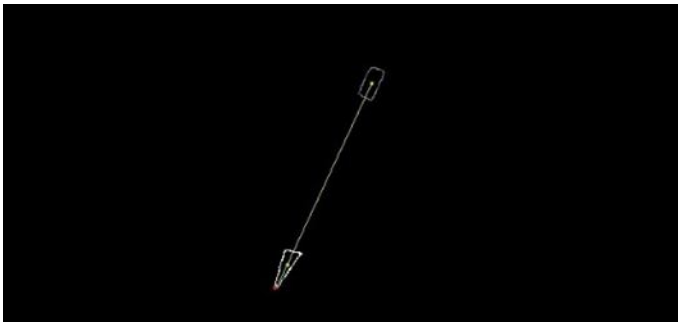


Figure 10: Detecting the pen tip

3.8 Finding pen tip 3D coordinates

By having the location of pen tip in both frames triangulation could be performed to get depth information. By knowing the focal length of the cameras and the baseline displacement from the calibration process carried out earlier 3D position can be found for the setup in Figure 11 as follow:

$$disparity = xl - xr \tag{9}$$

From equivalent triangles

$$z = \frac{f \cdot b}{disparity} \tag{10}$$

$$x = \frac{xl \cdot z}{f} \tag{11}$$

$$y = \frac{yl \cdot z}{f} \tag{12}$$

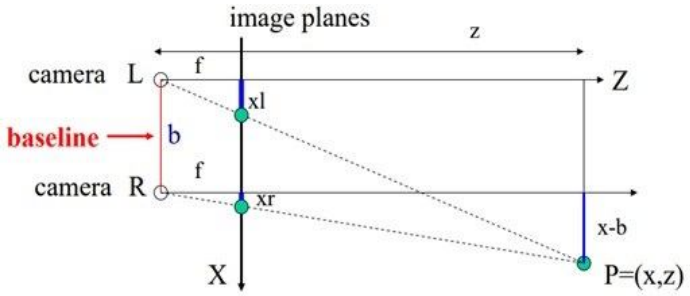


Figure 11: Top View of the stereo setup

3.9 Finding pen orientation

The orientation of the pen can be represented by 2 angles the altitude and azimuth as shown in Figure 12.

The altitude is the angle of elevation from the writing surface and the pen. While the azimuth is the angle formed between the pen projection on the writing surface and the positive of *y* axis. Knowing the points $p_1(x_1, y_1, z_1)$ and $p_2(x_2, y_2, z_2)$ the altitude and azimuth were calculated as follow:

$$Distance = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2 + (z_2 - z_1)^2} \tag{13}$$

$$Altitude = \sin^{-1} \frac{(z_2 - z_1)}{Distance} \tag{14}$$

$$Azimuth = \tan^{-1} \frac{(x_2 - x_1)}{(y_2 - y_1)} \tag{15}$$

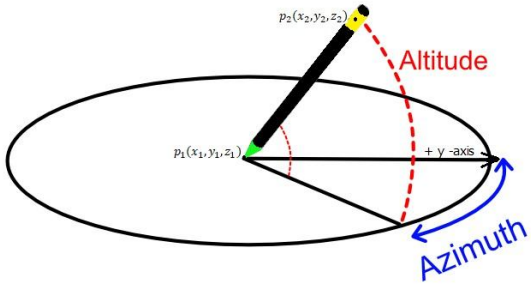


Figure 12: Pen Orientation Representation

4. TESTING AND EVALUATION

To evaluate the system two tests were performed one for the accuracy of the position tracking and the other is for the tracking speed.

4.1 System accuracy

To test the accuracy of the system, a device that has higher accuracy has to be used. Therefore, Wacom® Intous tablet was utilized for measurement. The colored markers were attached to Wacom stylus and the tablet is placed in front of the stereo camera. The tablet was connected to another computer to not inflict additional processing load on the main computer. Then some straight lines were drawn using a ruler along x axis and y axis. The length of the lines tracked by both the tablet and the stereo vision system were recorded. The results obtained were then compared and are shown in Table 1.

Table 1: Experimental Results Comparing Wacom Tablet Measurements to Measurements Obtained from the Stereo Camera

	Wacom Obtained Length (mm)	Camera Obtained Length (mm)	Error (mm)	Percent Error (%)	Distance to Stereo Pair (mm)
x (Depth)	52.75	53.23	0.48	0.9	250
	53	53.65	0.65	1.2	300
	51.50	52.36	0.86	1.7	350
	50.75	51.64	0.89	1.7	400
y	50.25	50.87	0.62	1.2	250
	51.50	52.22	0.72	1.4	300
	51.00	51.85	0.85	1.7	350
	50.75	51.63	0.88	1.7	400

The data collected in Table 1 indicated that the accuracy of writing data obtained by the stereo system had an error of less than 1mm when writing was within 400mm from the cameras. This is a significant improvement compared to the similar system using CPU by Fadi et al3 with image resolution of 640×480 pixels where error was in the range of 4mm within the same range from the cameras.

Following the initial test and additional visual test was conducted by drawing a shape on the tablet and tracking it in real-time using the stereo cameras. The graphs for the data collected from both systems were overlaid on a single plot shown in Figure 13. The data achieved align with the previous data as the error has increased as the distance *x* from the tablet increased.

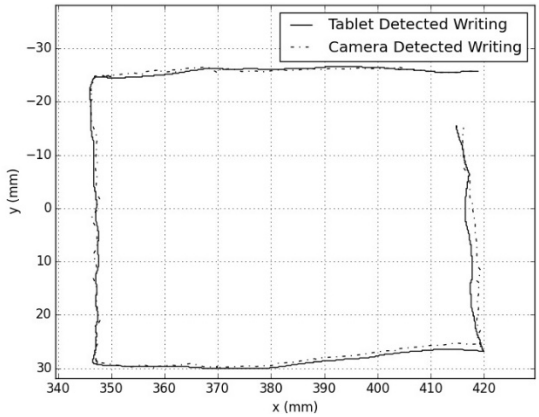


Figure 13: Tablet Position Versus Stereo Camera

After examining the system position detection accuracy, the tilt angle detecting accuracy was examined. Wacom® stylus was tilted back and forth while being tracked by the stereo camera. Then the data was collected and plotted in Figure 14. Since the tablet stylus has a maximum detection range between of 60º and -60º, it cannot detect over these limits. The data revealed that the system detection accuracy is within 5º of the value detected by the tablet.

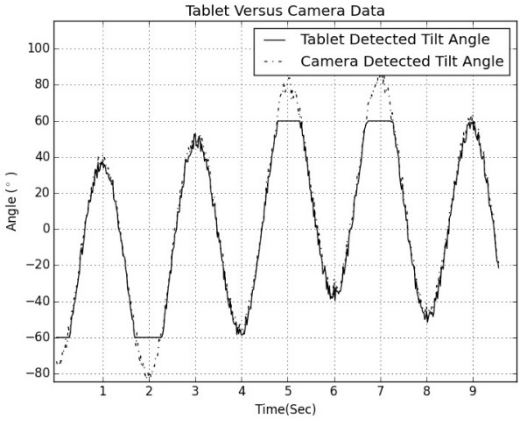


Figure 14: Tablet Angle Versus Stereo Camera

4.2 System tracking Speed

To evaluate the frame rate of the system different tracking algorithms were tried. Each tracking algorithm was run for 1 minute, the total number of frames were counted and divided by 60 seconds to find the average frame rate. The frame rate was tested for resolutions of 640×480 pixels and 1280×720 pixels.

Table 2: Experimental Results Comparing Frame Rate of Different Tracking Algorithms

Resolution	Processing on CPU [3]	GPU Processing with CCL on CPU	GPU Processing with Parallel CCL [8]	GPU Processing with 2 colored labels
640 × 480	14 FPS	22 FPS	28 FPS	30 FPS
1280×720	4 FPS	14 FPS	17 FPS	30 FPS

The results in Table 2 indicated that the algorithm developed could achieve processing at frame rate which is equal to the frame rate captured by the camera without any dropped frames. This could be achieved for high definition frames allowing for high accuracy and sample rate at the same time.

5. CONCLUSION

To make visual pen input systems a viable option compared to other input methods the accuracy had to be improved while maintaining the frame rate. This was possible only through parallel processing of images in GPU. Serial processing algorithms were ported to run in parallel on the GPU. The experimental results have proven that the system can achieve an error bellow 1mm at a frame rate of 30 FPS. The 30 FPS was achieved while some system resources are still available for additional post processing such as spatial resampling to compensate for device specific sampling frequencies and variations in the handwriting speed. This frame rate is still short of the current sampling rate of digitizing tablets which stands at about 200Hz. The 30Hz achieved is sufficient for most applications, however higher sampling rate is necessary for detecting applications with faster writing speed such as signatures.

REFERENCES

1. M. Moriya, T. Hayashi, H. Tominaga and T. Yamasaki, **Video tablet based on stereo camera - human-friendly handwritten capturing system for educational use**, Fifth IEEE International Conference on Advanced Learning Technologies (ICALT'05), pp. 909-911, 2005. https://doi.org/10.1109/ICALT.2005.303
2. Dave Jay, Venkatesh KS and Jain Garima, **Online 3D signature verification by using stereo camera & tablet**, proceedings of 23rd International Conference in Central European Computer Graphics, Visualization and Computer Vision, pp. 239-246, 2015.

3. Fadi Imad, Sharifah Mumtazah Syed Ahmad, Shaiful Hashim, Khairulmizam Samsudin, Marwan Ali, **Real-Time Pen Input System for Writing Utilizing Stereo Vision**, Journal of Computers, 13(9), 1000-1009, 2018.
4. G. A. Fink, M. Wienecke and G. Sagerer, **Video-based on-line handwriting recognition**, Proceedings of Sixth International Conference on Document Analysis and Recognition, Seattle, WA, pp. 226-230, 2001.
5. Hongjian Wang, Naiyu Zhang, Jean-Charles Créput, Yassine Ruichek, Julien Moreau, **Massively parallel GPU computing for fast stereo correspondence algorithms**, Journal of Systems Architecture, 65, 46-58, 2016.
https://doi.org/10.1016/j.sysarc.2016.03.002
6. Ge Li, Xuehe Zhang, Changle Li, Hongzhe Jin, Jie Zhao, **Design and application of parallel stereo matching algorithm based on CUDA, Microprocessors and Microsystems**, 47, Part A, 142-150, 2016.
https://doi.org/10.1016/j.micpro.2015.09.006
7. Aksoy, Yagiz; Aydin, Tunc; Pollefeys, Marc; Smolic, Aljoscha., **Interactive High-Quality Green-Screen Keying via Color Unmixing**, ACM Transactions on Graphics (TOG), 36(4), 2016
https://doi.org/10.1145/3072959.2907940
8. Ondˇrej ˇS´tava and Bedˇrich Beneˇs, **Connected Component Labeling in CUDA**, in *GPU Computing Gem*, Published by Morgan Kaufmann, ch35, pp 569-581, 2011.
https://doi.org/10.1016/B978-0-12-384988-5.00035-8
9. B. Gary and K. Adrian, ***Learning OpenCV: Computer Vision with the OpenCV***, 1st ed., Published by O'Reilly Media, ch12, pp 415-452, 2008.
10. Tsai, **An Efficient and Accurate Camera Calibration Technique for 3D Machine Vision**. Proceedings of IEEE Conference on Computer Vision and Pattern Recognition, Miami Beach, FL, pp. 364-374, 1986.
11. Z. Zhang. **A flexible new technique for camera calibration**. IEEE Transactions on Pattern Analysis and Machine Intelligence, 22(11), pp.1330–1334,2000.
https://doi.org/10.1109/34.888718
12. Ramesh Jain, Rangachar Kasturi, Brian G. Schunck, ***Machine Vision***, Published by McGraw-Hill, ch11, 1995.
13. P. Sebastian, Y. V. Voon, and R. Comley, **Colour space effect on tracking in video surveillance**, International Journal on Electrical Engineering and Informatics, 2, 298-312,2012.
https://doi.org/10.15676/ijeei.2010.2.4.5
14. Amir Hesam Yaribakht, Mohd Shahidan Abdullah, Alireza Ghobadi, **A Novel Color Image Watermarking Method based on Digital Wavelet Transform and Hungarian Algorithms**, International Journal of Advanced Trends in Computer Science and Engineering. Volume 8, No.2, pp.154–164, 2019.
https://doi.org/10.30534/ijatcse/2019/09822019
15. Llorente, Cesar & Dadios, Elmer, **Development and characterization of a computer vision system for human body detection and tracking under low-light condition**. International Journal of Advanced Trends in Computer Science and Engineering. Volume 8, No.2, pp.251–254, 2019.
https://doi.org/10.30534/ijatcse/2019/24822019