Improvement of Vibration and End-to-End Latency for Optical Motion Detection System in Magnetic Resonance Imaging

Yifan Wu

A thesis

submitted in partial fulfillment of the

requirements for the degree of

Master of Science in Bioengineering

University of Washington

2017

Reading Committee:

Chun Yuan, Chair

Niranjan Balu

Gador Canton

Program Authorized to Offer Degree:

Bioengineering

University of Washington

Abstract

Improvement of Vibration and End-to-End Latency for Optical Motion Detection System in Magnetic Resonance Imaging

Yifan Wu

Chair of the Supervisory Committee:
Professor Chun Yuan
Radiology and Bioengineering Department

This dissertation presents improvements to both software and hardware for a marker-less optical motion detection system used in magnetic resonance imaging. Multiple devices have been developed to eliminate artifacts in MRI caused by patient movement, as motion is a common cause of artifact. Most developed devices require attaching pieces to a subject's body; however, our optical motion detection system is attachment-free. Previously, we successfully reduced imaging artifacts retrospectively using carotid artery MR data. We used gathered motion data to determine beginning and ending times of the movement, then discarded the corresponding k-space lines within the movement duration to eliminate artifacts. Now we would like to push this process from retrospective correction to real time. However, the camera and laser mounting vibration, as well as the system time delay variation, generate inaccuracies which make it difficult to find the exact k-space lines to be removed. To make the artifact-eliminating process more efficient, we reduced

the overall end-to-end latency from 110ms ± 21ms to 100ms ± 12ms by coding the motion processing program in Python with OpenCV library. We also designed a new laser and camera holder using 3D printing with PLA for better stability. The noise and drift in both x and y directions are decreased with the new holder compared to the previous setup. The same result for decrease in vibration was seen during phantom scanning with T1-weighted (TR = 100ms) gradient echo (GRE) and with T1-weighted turbo spin-echo (TSE) quadruple inversion-recovery (QIR) (TR = 800ms) sequence. Furthermore, the 3D printed holder decreases the overall assembly time required to set up the system. Overall, the new design of the system reduces the event-to-display latency, the latency variation, the hardware vibration, and the assembly time. The new design improves the old design and provides a path to pursue prospective optical motion correction.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGEMENTS

I would like to greatly acknowledge all the people in the Vascular Imaging Laboratory in the Department of Radiology at the University of Washington for helping me completing this thesis. Particularly, I would like to thank my advisor, Dr. Chun Yuan, for his extensive guidance on this interesting project. It was an honor to work with a renowned expert in the magnetic resonance imaging field. I also would like to thank my other two committee members, Dr. Gador Canton and Dr. Niranjan Balu, for their kind and endless help with my research. Moreover, the project would not be possible without equipment and manpower support from Comotion MakerSpace in the University of Washington. Finally, I would like to thank my mentor, PhD candidate Jin Liu, for providing tremendous assistance on imaging experiments and overall project design.

I would like to conclude acknowledgements with great thanks to my parents, for their warm love and support during my studies.

To my parents

Chapter 1. INTRODUCTION

1.1 MOTIVATION

Patient movement is the major cause of artifacts in magnetic resonance imaging (MRI). In the year 2014, the economic loss due to artifacts caused by patient motion in the US was approximately $1.4 million [1, 2]. Artifacts are generated if there is a corrupted k-space line, and the artifacts caused by these k-space lines will be reflected on the whole image [3]. Many techniques have been previously developed to eliminate artifacts caused by motion. These techniques can be loosely divided into two categories: navigator-based techniques and external device-based techniques.

Navigator-based techniques include orbital Navigator Echo (ONAV) [4-7], motion reducing K-space encoding/reconstruction [8], and Periodically Rotated Overlapping ParallEL Lines with Enhanced Reconstruction (PROPELLER) [9, 10], etc. These techniques aim to reduce ghost and blurring artifacts within scanning by modifying the pulse sequences. They do not need external devices but require a significant increase in scanning time, and only partial artifacts can be eliminated.

External device-based motion detection techniques, such as the breathing belt and optical marker tracking methods, require pieces attached to a patient's body in order to track movements. The breathing belt aims to monitor diaphragm position and uses these diaphragm traces to control the scanner to accept data only at the breathing gap between inspiration and expiration. Thus, the scanner only acquires signals when the diaphragm lies within a certain acceptance range [11]. Optical marker-tracking techniques have been proved as an acceptable way for measuring motions [12]. These techniques, however, will cause discomfort due to the external attachments on subjects.

Previously, our lab has developed a marker-less device-based optical motion detection system that uses structured light and MR-compatible camera to detect subject's motion and can eliminate resulting artifacts after post-processing work. We used our collected motion data to find time slots that involve movements, and we discarded the corresponding corrupted k-space lines. The artifacts are eliminated by estimating and reconstructing the deleted k-space lines from their neighboring lines. This previously designed system can successfully gather respiration, random, and bulk motion of subjects, and we used neck area as a window to detect movement. However, the mounting of the previous design added vibration noise that decreases the accuracy of determining the exact motion time and lowers the efficiency of k-space reconstruction. Moreover, the image artifact reduction is based on retrospective correction, and there is uncertainty in the time delay variation from motion to computer motion-recognition, which also decreases system efficiency. In order to efficiently eliminate imaging artifacts retrospectively and prospectively, mounting vibration needs to be decreased and the duration between the actual movement occurring to computer motion recognition needs to be minimized.

1.2 Outline of the Dissertation

This dissertation is focused on improving the motion detection process of an existing optical marker-less MR-compatible movement-detecting system, and therefore it is focused on the re-design of 1) the mounting of an established optical motion detection system and 2) the software coding that reduces the delay between camera signal detection and computer motion recognition. The main contribution of this dissertation can be summarized as:

- A hardware design that shortens total assembly time, decreases vibration, and is adaptable for all MRI coils

- A software design that reduces event-to-display latency between camera and computer motion-recognizing
- A linear-fit statistical method using residuals that proves the reduction of vibration
- A comparison of standard deviation and mean value between previous and new software design that proves the reduction of time delay

The organization of this dissertation is as follows. Section 2 discusses the motions involved in MRI and solutions of those motions. Section 3 explains the design of system hardware and software. Section 4 describes the end-to-end latency reduction test and results. Section 5 expresses the vibration reduction test and results. Finally, Section 6 addresses the possible ways to apply this system on MRI scanning, as well as conclusion and future approaches.

Chapter 2. MOTION

2.1 Different Types of Motion

We focus on three different types of motion: respiratory, random, and bulk motion. Respiratory motion is defined as motion caused by breathing, and is periodic, as well as predictable. Since we use the neck to detect a subject's movement, the random motion we are interested in is un-predictable motion that could happen in the area between head and trunk, such as coughing and swallowing. Random motion lasts only a short amount of time. Bulk motion in this case is defined as slow, non-periodic movement, such as drift or head turning movements.

2.2 MOTION DETECTION

2.2.1 *Camera and Laser Setup*

Fig. 2.1 shows the camera and laser setup we used for detecting patient's neck motion. We used a structured light (as shown in Fig. 2.2) to shine on subject's neck, and a camera focusing on the same plane with the laser to receive motion signals. The distance between camera and laser is represented as D, and the distance between camera/laser plane and reference plane (bed) is H; based on trigonometry, the height of the subject, h, can be calculated using the equation below:

$$h = \frac{dkH}{D+dk} \tag{2.1}$$

where k is the length ratio between real object and image, and d is the laser position in the image plane of the camera[12].

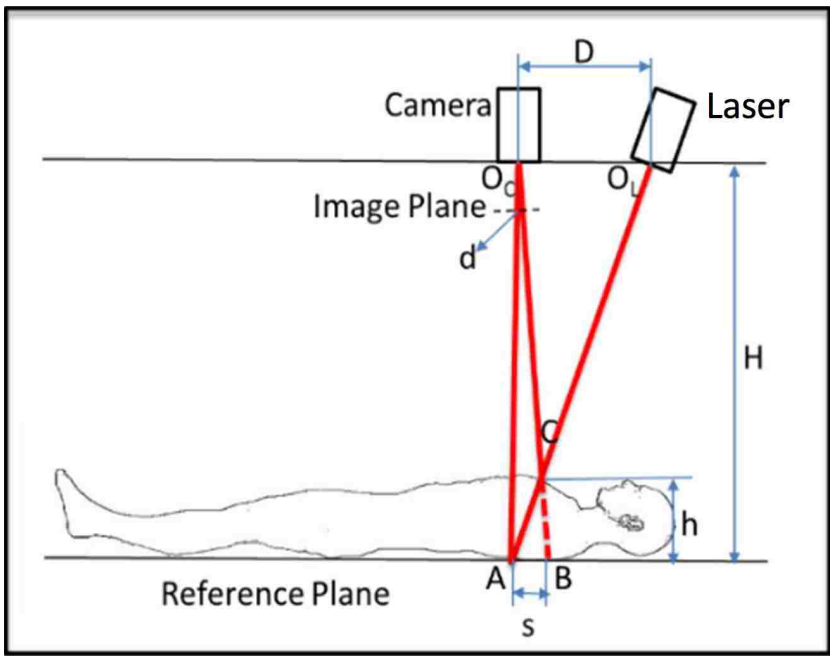


Figure 2.1 Optical motion tracking system. Camera can capture the projected laser shift caused by the height (h) changing of patient [12]*

Figure 2.2 Structured light on volunteer's neck

2.2.2 *Motion Data*

The camera records a 2D graph of the structured light position, and an (x,y) coordinate is used to represent the center of the light. Since the three types of motion majorly occur along the head-toe (y) direction, a graph of y-direction vs. time is used to show the detected movements. Fig. 2.3 illustrates the respiratory movement, random/abrupt movement (cough) recorded from a healthy volunteer. The black line is the raw data acquired from our system, and the blue line is the reference breathing data from breathing belt. A significant change in frequency can be noticed when the random motion occurs. Moreover, a slight drift can be seen after coughing (dashed red line). The drift is calculated based on the mean of y-position during the respiration period before and after the random movement [12].

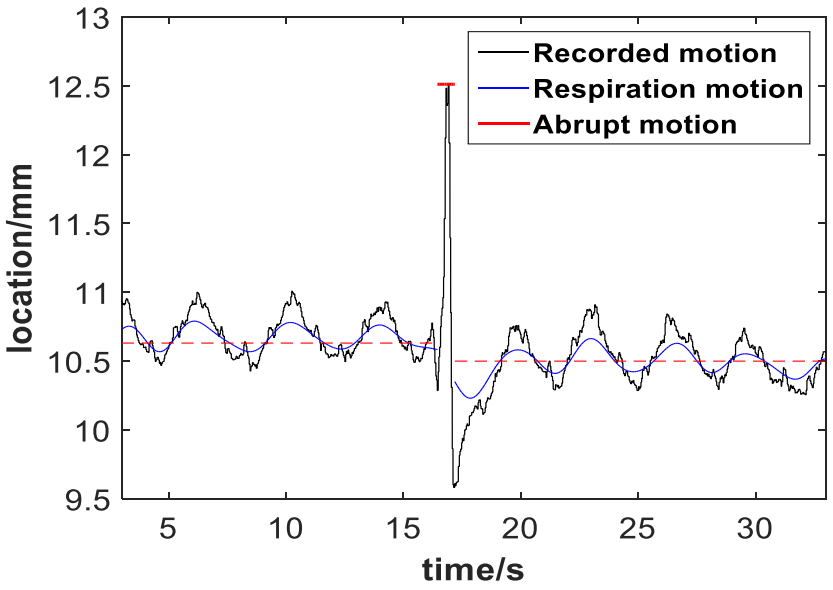


Figure 2.3 Marker-less optical remote motion sensing (black curve) can detect respiration motion accurately and robustly with frequency highly correlated with respiration belt (blue curve) [12]*

2.3 Motion Correction (Retrospective)

The retrospective motion correction was developed by deleting corrupted k-space lines. Based on the motion signal acquired, random motion (cough) period can be defined using the time stamps collected with the movements. After eliminating the corrupted k-space lines, the missed lines are reconstructed using established parallel imaging algorithms. Bulk motion (drift) is also corrected by reconstructing k-space lines before and after coughing. Fig. 2.4 shows the result of retrospective correction of coughing on carotid imaging [12].

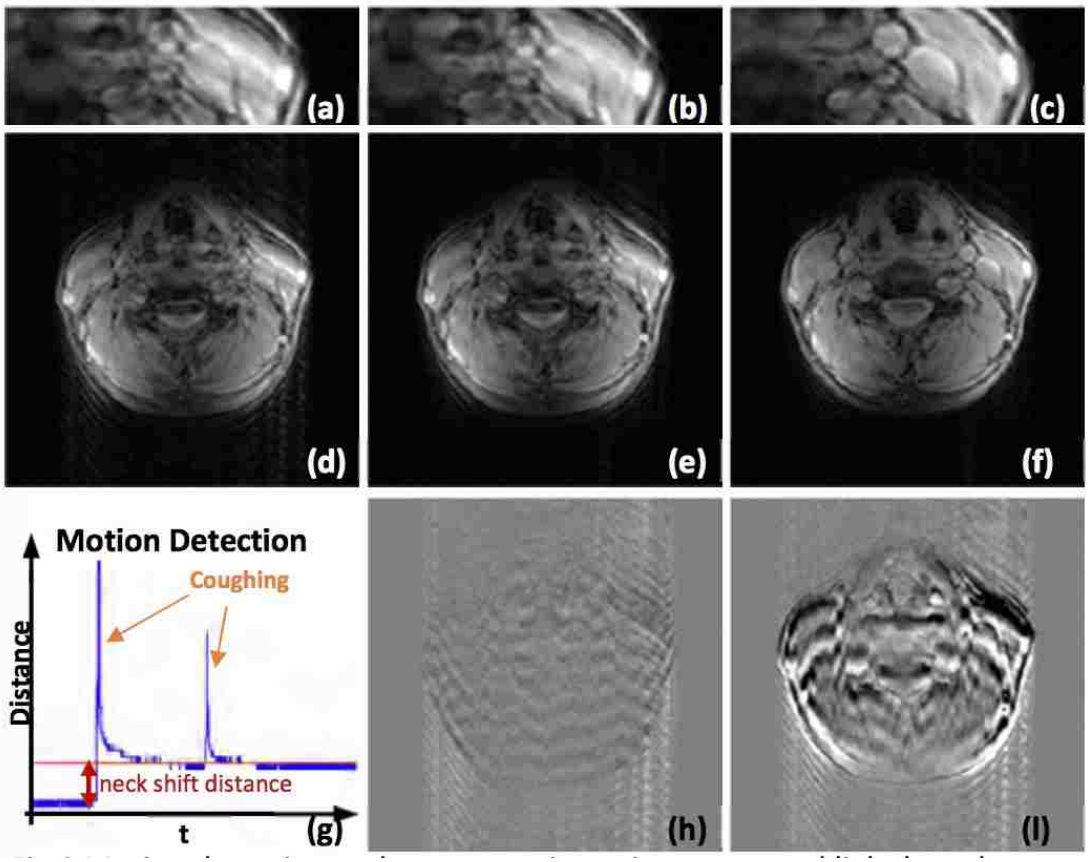


Figure 2.4 Motion detection and compensation. Images (a)-(c) are the zoomed carotid artery images of the original scan (a), after removing corrupted data (b) and after removing corrupted data and compensating neck shift (c); images (d)-(h) are the corresponding full FOV; (g) represents the motion recorded by the structured light system; (h) captures the difference between (d) and (e); (l) captures the difference between (d) and (f) [12]*

Chapter 3. SYSTEM DESIGN

The system uses structured light and an MR-compatible camera to capture motion, the corresponding signal is then sent to a computer for motion recognition. Thus, the overall system includes two major components, hardware and software. The hardware includes all the equipment involved in the motion detection, and software includes all the procedures for computational processing of movement recognition.

3.1 HARDWARE DESIGN

Hardware includes the MR compatible camera, connection cables that transmit data from camera to computer, structured light laser, and 3D printed holder for supporting camera and laser.

3.1.1 *MR Compatible camera*

Since we propose to detect patients' motion during MR scanning, a MR-compatible camera is needed. We used 12M high-speed camera from MRC, Germany. The camera has 30hz frame rate and includes a color sensor. The focal length of this camera is selectable, and the working environment of this camera requires a filter box which has optical isolation of video signal, a connection cable, a power supply, a BNC cable, and a BNC adapter to transfer image signal from the camera to the computer. The working pathway is showed in Fig. 3.1.

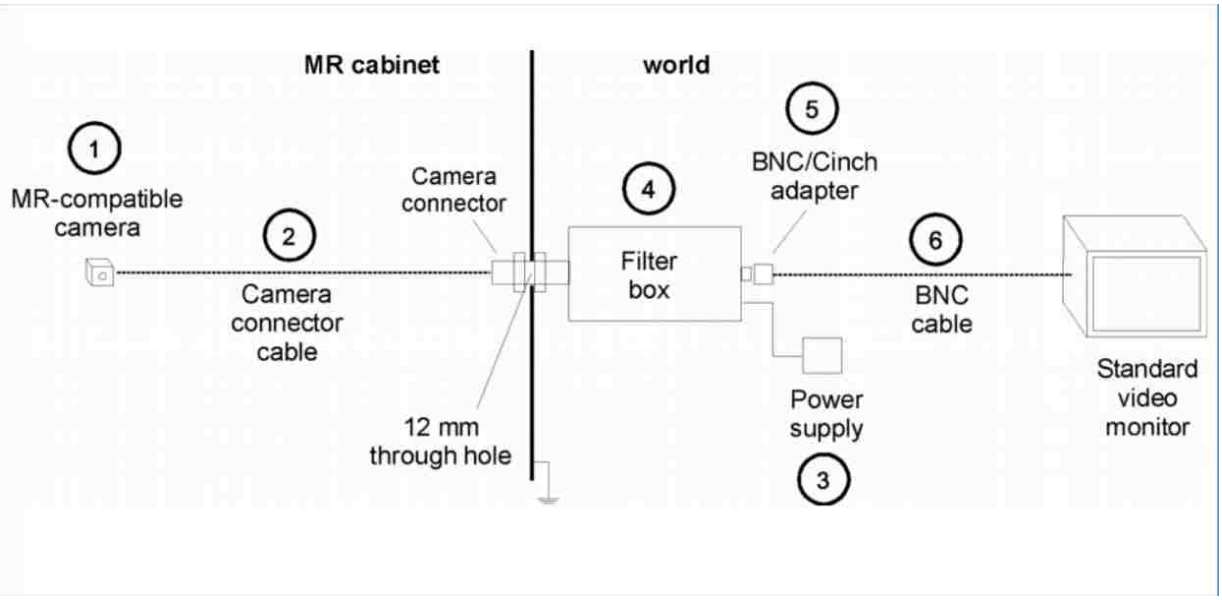


Figure 3.1 Device and figure configuration of MR compatible camera working pathway [13]

We used a Dell laptop as a standard video monitor, and since the BNC cable cannot directly connect to the laptop, an analog to digital (A/D) USB adaptor was used. August VGB100 Video

Capture Card was the adaptor that connected the laptop and the BNC cable. Due to the nature of USB adaptor, adding this adaptor will result in increasing the variability of overall time delay.

3.1.2 *Structured Light Laser*

We used a 532nm wavelength (green light) GM-CR02 laser module to shine two bright crossing lights onto subject's neck (Fig. 3.2). The operation voltage for this laser is 3V, DC.

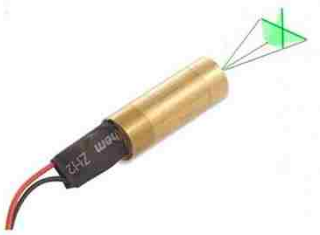


Figure 3.2 CM-CR02 laser module

3.1.3 *3D Printer and Material*

A FlashforgeUSA 3D Printer with 0.1mm nozzle was used to print the laser and camera holder. The 3D printer has a maximum printing space of 22.5cm (length) × 14.5cm (width) × 15.0cm (height) and is provided by Comotion MakerSpace in University of Washington. Polylactic acid (PLA), which has a melting point between 130°C – 180°C, was chosen as the printing material simply because it is widely available and resistant enough to support all the weight [14]. The software incorporated is MakerBot Print, and the files were printed directly from the SD card.

3.1.4 *Camera and Laser Holder*

As mentioned previously, the holder is used inside the MR scanning room, so there should be no metal involved, and PLA is an appropriate printing material. A clear plastic table was made to place on the scanner bed, so that the patients can lay between this table and scanner bed. The holder is placed on the table to hold camera with laser in order to form the combination as shown in Fig. 2.1.

SolidWorks was used to 3D design the holder. The overall looking of the holder assembly and individual pieces are shown in Fig. 3.3. there are four pieces of the holder, the base, the camera piece, the camera-base connection, and the laser piece. Plastic bridge (in Fig. 3.3 a)) measurements are in inches, and all other lengths (Fig. b) – h)) are in millimeters. English units are used for plastic bridge measurements because this specific design part was handmade by staff in a plastic shop, where only English units could be taken. Whereas the 3D printer can only read metrics.

To make the assembly capable for volunteers of different sizes, we added an extension for the camera piece. Moreover, a supporter is added to stabilize the extended camera. In this way, the monitor area of the camera is lengthened, and the overall assembly is adaptable for scanning with a head coil, carotid coil, and abdominal coil. The extension pieces with supporter and assembly is illustrated in Fig. 3.4.

Figure 3.3 Individual pieces of the design and overall assembly: a) camera piece b) plastic bridge c) the base d) base-camera connection e) laser piece f) overall assembly without plastic bridge g) assembly with plastic bridge h) actual assembly looking (with camera and laser)

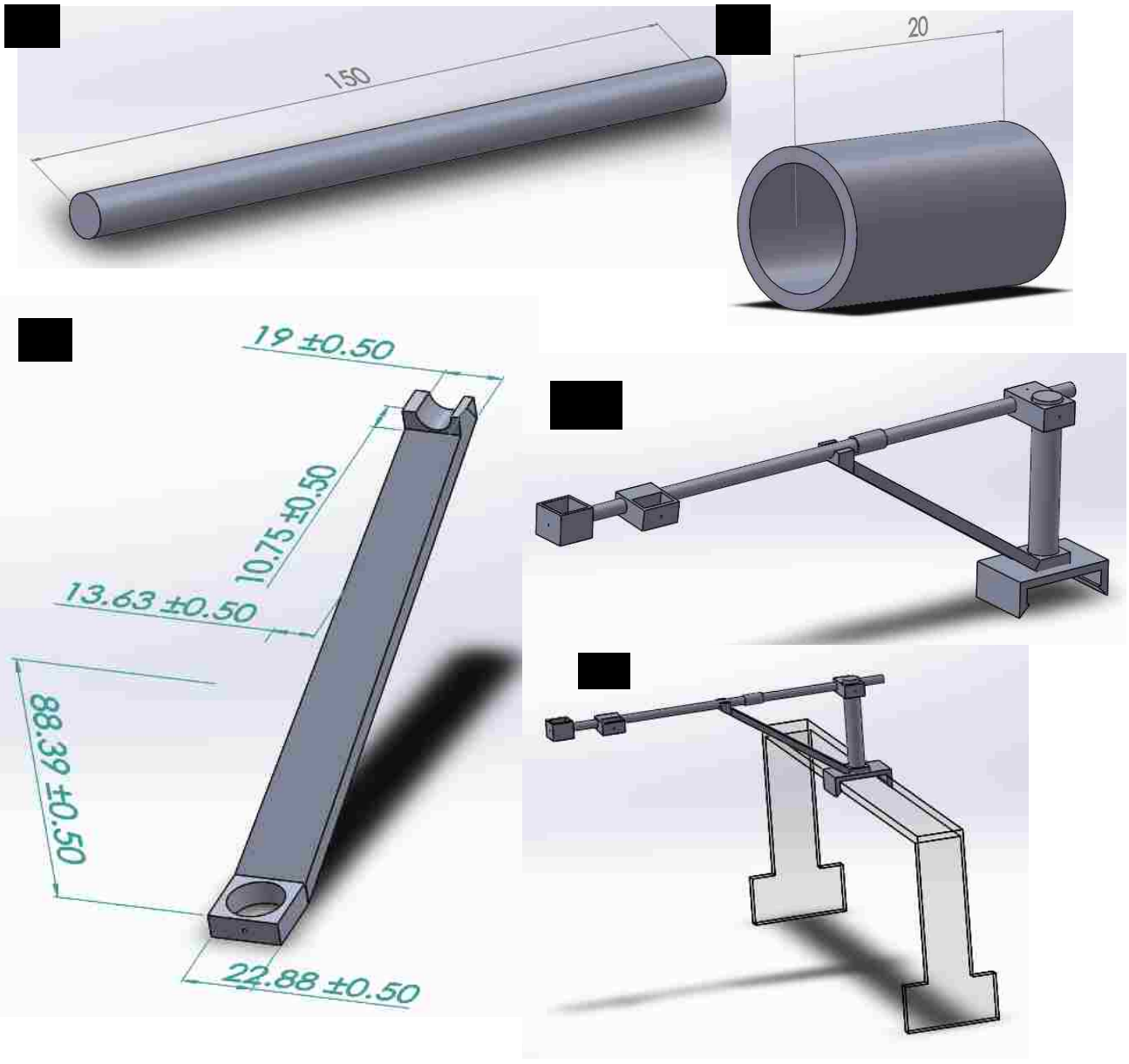


Figure 3.4 Extension pieces and assembly with extension. a) extension rod b) extension rod buckle c) extension supporter d) assembly of extension pieces without bridge e) assembly of extension pieces on plastic bridge. All measurements are in millimeters.

3.2 SOFTWARE DESIGN

3.2.1 *Importance of time delay and delay variance*

Previously we determined corrupted k-space lines using collected motion data. We synchronized the computer system time clock with the scanner, so that we could determine the corresponding k-space lines during the period where the motion happened [12]. The beginning and ending time points of motion are determined by coding software using the movement data shown in Fig. 2.3. Since there is latency variation between when motion event occurs and when the software recognizes the motion, the determined motion-happening time is always delayed with respect to the actual event-happening time, and this time delay varies. Knowing that, when discarding k-space lines, we must edit more k-space lines due to the delay variance. The number of additional k-space lines to be deleted are determined based on the repetition time (TR), which represents the time it takes to draw a single k-space line. The TR time differs from a few milliseconds to hundreds of milliseconds [15]. Thus, minimizing the overall system time delay and delay variance in the scale of milliseconds is vital for efficiently eliminating artifacts in real-time. The goal of delay variance reduction is 3ms, which the usual shortest TR time in MR scanning; to approach that goal, we replaced graphic programming language with object-oriented programming language.

3.2.2 *Challenge of LabVIEW*

Previously we used LabVIEW to monitor and record the motion captured by camera. Although it is easy to use and code, LabVIEW is a graphic programming language, the time it takes to run through every loop is not exact leading to a time variation in the execution of a given code each time it is run. Thus, we proposed to change the programming language from graphic programming

language to object-oriented programming language. We chose Python instead of LabVIEW to monitor and process the motion data.

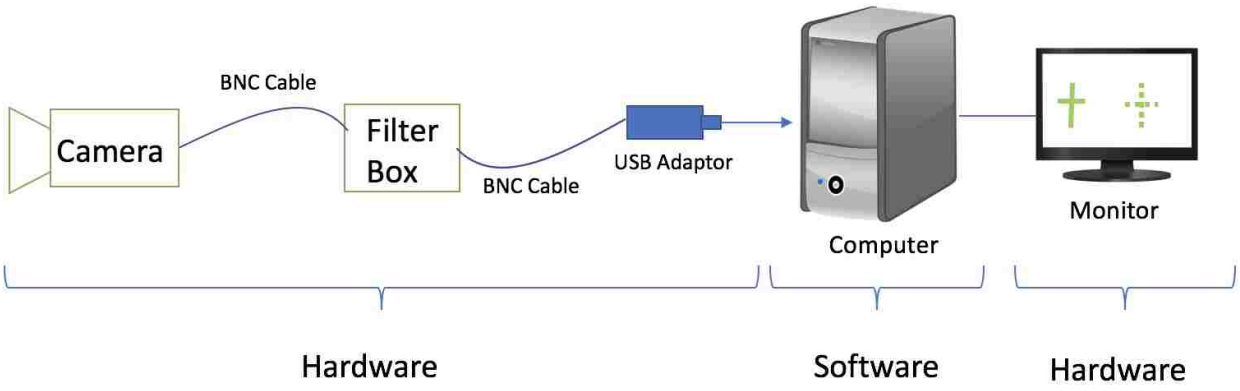
3.2.3 *OpenCV Library*

Since the image captured is a colored 2D frame, OpenCV 2.0 library is used to process the motion data in real time. OpenCV is a library widely used for computer vision and machine learning. With more than 2500 optimized algorithms, OpenCV is used to recognize the structure light pattern and record the relative position of the pattern's center. OpenCV has C++, C, Python, Java and MATLAB interfaces [16]. Python is used due to its simplicity and free open-source nature.

Chapter 4. END-TO-END LATENCY REDUCTION

4.1 COMPOSITION OF DELAYS

The end-to-end latency refers to the time delay between the actual time when motion occurs and the time when the motion is recognized by the computer and is displayed on the computer monitor. Therefore, any components between the subject and the computer screen that are responsible for data transmission and processing will contribute to the overall time delay. The latency components are shown in Fig. 4.1.



4.1.1 *Hardware Delay*

The MR compatible camera has a frame rate of 30Hz, thus the time needed to capture each frame is approximately 30ms; that is, there is a 30ms delay between every frame. We can infer that if the software processing time is less than the frame rate, an upgrade on higher speed camera can guarantee the reduction of end-to-end delay.

Referring to Fig. 3.1, the cables and the filter box all contribute to the time delay; moreover, we used a USB adaptor to connect the BNC cable and the computer, which will increase the total time delay. The monitor screen of the computer we used has a refreshing rate of 60Hz; thus, the duration of each screen frame is 1/60s (16.67ms). Therefore, there is a time gap (up to 16.7ms) between the processed movement decision and the actual display. The Windows System operation time, which adds into the overall latency, also partially depends on the build-in hardware of the computer, but the delay caused by hardware is non-modifiable. Reducing hardware delay would imply to change to a different and more efficient computer; thus, reduction of this delay is not discussed here.

4.1.2 *Software Delay*

The software delay is considered as the time the USB adaptor sends the first signal until the event is displayed on the computer screen. Here we define several time points that represent software time delay components, and all these delays occur inside computer and are due to the different image processing steps within the computer. The division of these components is illustrated in Fig. 4.2.

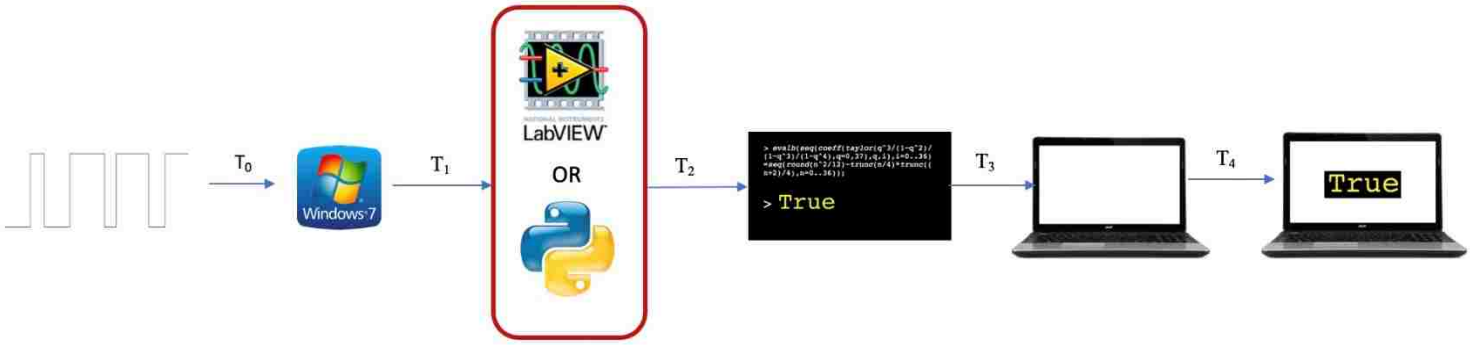
T_0: Digital signal represents last pixel of the picture received

T_1: Proper application is chosen to process the received signal

T_2: Motion detecting decision is made based on processed signal

T_3: Computer screen receives the decision

T_4: The decision is displayed on the screen



The time delay from T_0 to T_1 and from T_2 to T_3 depends on the operating system and operation environment; the latency between T_3 and T_4 depends on the monitor refreshing rate. The processing environment is Windows 7 enterprise 64-bit with Intel Coe i7-4810MQ CPU at 2.80GHz. All irrelevant applications are closed when running the experiments to ensure the delay components of T_0- T_1 and T_2 – T_3 are the same every time. Here we focused on reducing the time between T_1-T_2 since this is the time duration we can control.

4.2 Delay Measuring Method

4.2.1 *Delay Measurement Logic*

We propose to measure the delay without any usage of external devices. Therefore, we use a measuring method similar to what Gullichsen illustrated in his thesis [17]. We used computer screen to mimic the motion of the structured light (green cross) showed on patient's neck. That is,

we displayed a green cross and programmed the cross to move every second. Then we connected the MR compatible camera the computer and monitor the green cross movements on screen. The logic is graphed in Fig. 4.3. The computer is separated as monitor and mainframe for better illustration. The computer only processed the graphic image gathered from camera; thus, it only read the motion from the camera, and the moving green cross displayed on the screen was ignored by the motion detecting program. In this way, the actual event-happening time point (when the cross moves) and the event-recognizing time point (when the computer finishes processing the motion from camera) could be accurately recorded, since only one computer was involved in the whole progress.

Again, we define several delay components to make the process clear:

T_0: Preparation of event starts

T_1: Moving event sends to screen

T_2: Moving event shows up on screen

T_3: Camera receives the last pixel of image

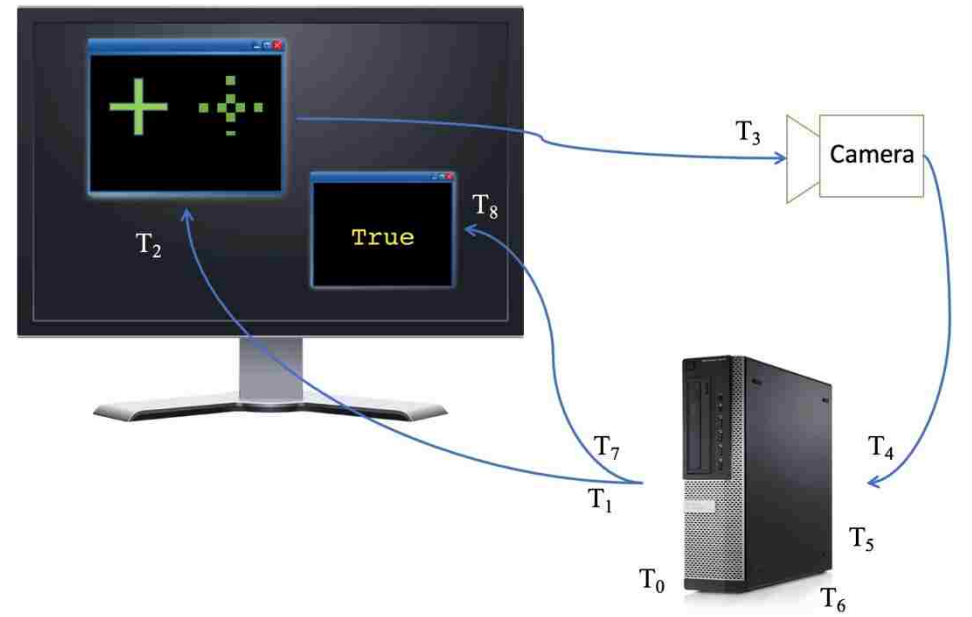
T_4: Last signal of picture received by computer

T_5: Proper application is chosen to process the received signal

T_6: Motion detecting decision is made based on processed signal

T_7: Computer screen receives the decision

T_8: The decision is displayed on the screen



A simple graphic program was written to generate a moving green cross on the screen, and that would move every one second. The camera captures the motion on the screen and sends the signal to the computer to be processed. The computer then only processes the signal sent from camera and determines if there is a movement. This decision is displayed on the same screen for us to judge. The camera is placed close enough to the screen so that the decision information is not seen by the camera, and the only pattern the camera can capture is the green cross.

The delay component we can control and are interested in is $T_5 – T_6$. Although the movement of the green cross is controlled by a code, the process of recognizing motion is independent from the code controlling the pattern movement. The motion recognition process only captures movements from camera, so that $T_7 – T_8$ does not contain T_1- T_2.

4.2.2 *Programming Algorithm of Delay measurement*

The background of the moving green cross was set black to remove interference fringes from the screen (shown in Fig. 4.4). As mentioned before, OpenCV library was used to process the motion signal. A while loop was used to continuously capture images from the camera; the main program loop consists of the following structures:

`cv2.Videocapture()`: This function is called once to open the camera shutter and begin gathering pictures

`cv2.cvtColor()`: This function converts the color image to grayscale for later filtering out the green color shape we are interested in.

`cv2.threshold()`: This function creates binary threshold images ready to be filtered

`cv2.inRange()`: This function creates a shape mask that filters out the pattern that we are interested in

`cv2.drawContours()`: This function draws contours of the masked-out pattern

`cv2.moments()`: This function calculates the momentum center of the contour, and the coordinates of momentum center is the position we monitor for movement.

`time.time()`: This function returns the current time in seconds, and we use the returned value multiplied by 1000 to get current time in milliseconds.

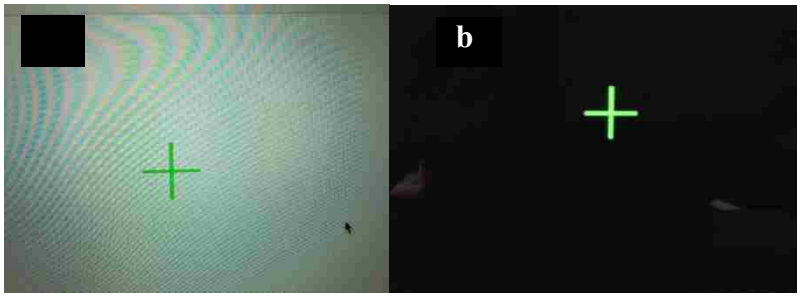


Figure 4.4 Appearance of pattern under different background. a) interference fringes b) no fringes with black background

The purpose of the main loop is to read pictures from the camera continuously and record the time point once the momentum center of the recognized pattern changes its position. The overall code can be found in the Appendix. There is noticeable noise of the recorded pattern; thus, recording any changes of the center coordinates renders false positive results. Therefore, calibration is needed. The calibration aims at identifying noise when the pattern is not moving. We are only interested in the relative position of the momentum center since we only want to record the change in position, so matching the recorded image to screen coordinate space is not needed. We recorded the position of the momentum center when it is not moving for 2 minutes, and the changes in the coordinates are the noise level.

4.3 RESULTS

The noise we recorded was 1 pixel, so we set the program to read changes when the momentum center of the pattern moves 2 pixels. Previously the delay was measured using the same logic but with LabVIEW software. Table 1 shows the comparison of results for 25 experiments.

	LabVIEW	Python
Average time delay	112ms	100ms
Delay standard deviation	21ms	12ms

Table 4.1 End-to-end latency comparison between LabVIEW and Python

The distribution of the delay between the two software is illustrated in Fig. 4.4. All data are normally distributed and a significant difference in both average and standard deviation can be seen. Standard deviation represents the stability of the processing time. There is bigger variance in LabVIEW than in Python.

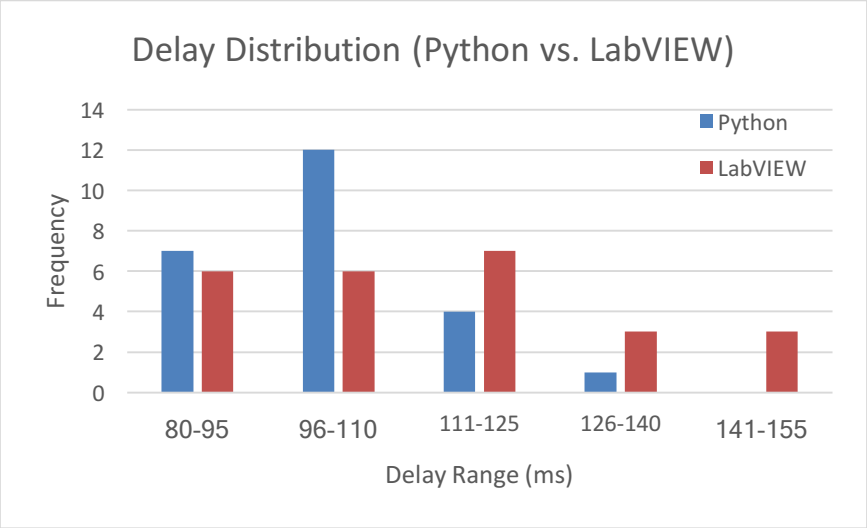


Figure 4.5 Delay distribution of Python (blue) and LabVIEW (red)

4.4 Discussion

As mentioned earlier, Python is an object-oriented programming language and is faster than LabVIEW. Although LabVIEW is easier to use for scholars without much programming experience, a more efficient method should be chosen to achieve real-time motion recognition. Python has characteristics of faster processing time and less variance, and therefore it should be used for future motion detecting work. LabVIEW, on the other hand, is less efficient and less stable. In LabVIEW, time varies every time even when it runs the same program. Moreover, LabVIEW is not an open-source software; thus, we could not measure the time needed for each step and identify the largest variation during the process. This problem is avoided when using Python since we can write the code and monitor the running time for each line of the code. The instability of the motion detection system is vital since we want to record the exact time of motion occurrence period and determine the correlated abrupt k-space lines to reduce artifact[12]. Large standard deviation will result in tracing back more k-space lines, which can result in longer scanning time or harder work on artifact elimination based on different image

correction applications. The overall delay calculated here includes all the delay components illustrated in Fig. 4.2, but some of the components, such as the screen refreshing delay, are not involved in the optical motion detection system when recording patients' movements. We can infer that for testing on real subjects, the average and standard deviation of end-to-end latency is shorter than tested result addressed above.

Chapter 5. VIBRATION REDUCTION

5.1 METHOD

Previously we used a PVC pipe to mount the camera and the laser. The pipe was long enough so that it could reach the middle of the main coil, and the end of the pipe that was outside the main coil was connected to a vertical pipe that could stand on the ground. Here we compare the vibration noise between the new mounting design (3D-printed holder) with this old design.

We used a 2000cc 3T MRI phantom bottle filled with Spectrasyn 4 to measure the vibration of the system setup during scan. We scanned the steady phantom using the 3D printed holder and the previous holder design, and for each design the phantom was scanned three times, one without adding any gradient (no grad), one with T_1-weighted gradient echo (GRE) sequence (TR = 100ms), and one with T_1-weighted turbo spin-echo (TSE) quadruple inversion-recovery (QIR) (TR = 800ms). The scanner itself will vibrate during scan, and that vibration will affect the holder stability. The more violent the sequence is, the bigger the vibration will be. The GRE sequence has lower gradient and less flip angle compared to TSE QIR sequence, since QIR sequence has two double-inversion RF pulse pairs [18]. Therefore, more vibration will be generated by the scanner when the turbo spin-echo QIR sequence is used. A ruler was placed on the phantom for calibration (Fig. 5.1). Since the movement caused by vibration was small, the

distance of the green cross movement was approximately parallel with the camera-laser plane shown on Fig. 2.1. The camera resolution is 720 × 480 pixels; therefore, we can easily calculate the relationship between moving distance of the green cross on phantom and pixels captured by the camera.

The GRE sequence and TSE QIR sequence are two different sequences with different gradients. We would like to know, under the GRE sequence, if higher gradient will cause more serious vibration. We modified only the gradient for another GRE sequence and scanned the phantom 6 times with 5, 10, 15, 20, 25, 30 mT/m. The results are shown in Table 5.1.

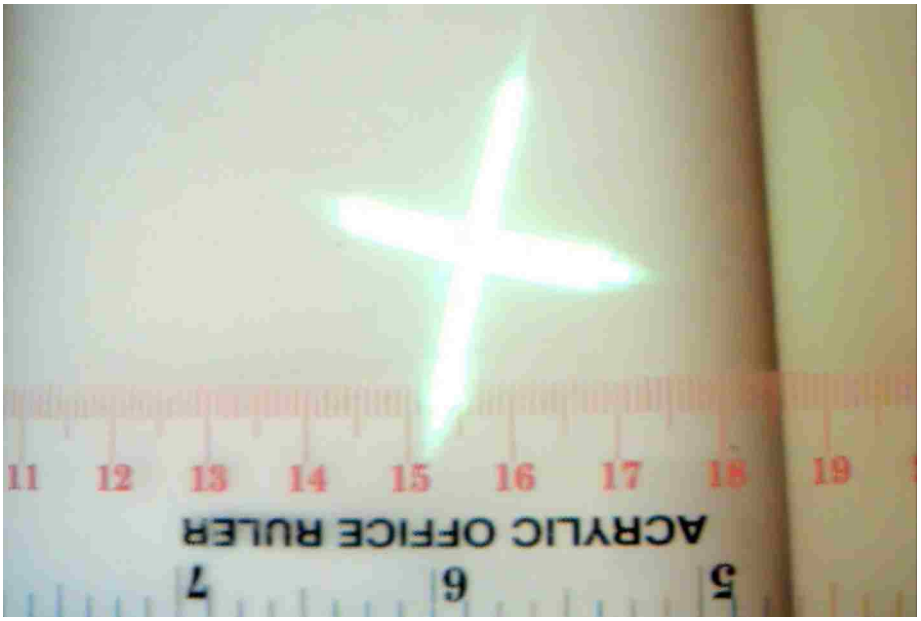


Figure 5.1 Calibration view from the camera.

5.2 Result and Discussion

The vibration of the new 3D printed camera holder is less for all phantom experiments in all three conditions. A linear fit is drawn for each graph in consideration of uniform drift during experiment. The residuals then indicate the vibration of the device without drift, and the norms of residuals represent the scale of vibration. The size of 1 pixel relative to the green cross moving

distance is 1 pixel = 0.127mm. The results are shown in Fig. 5.2 (no gradient added), Fig. 5.3 (phantom scanned with T_1-weighted GRE sequence), and Fig. 5.4 (phantom scanned with T_1-weighted TSE QIR sequence).

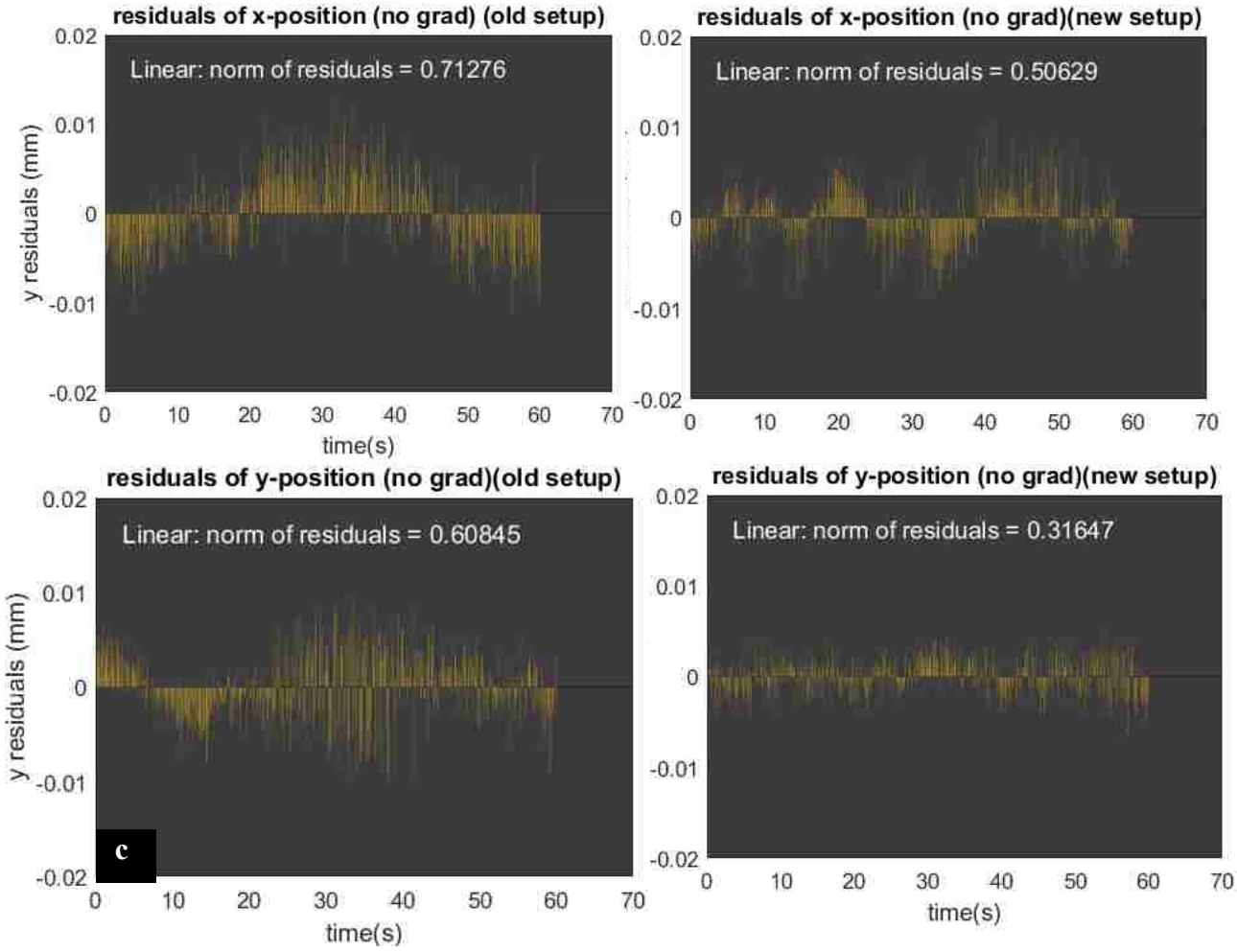


Figure 5.2 X and Y positions of recorded moving pattern without scan. a) X position of old design b) X position of new design c) Y position of old design d) Y position of new design

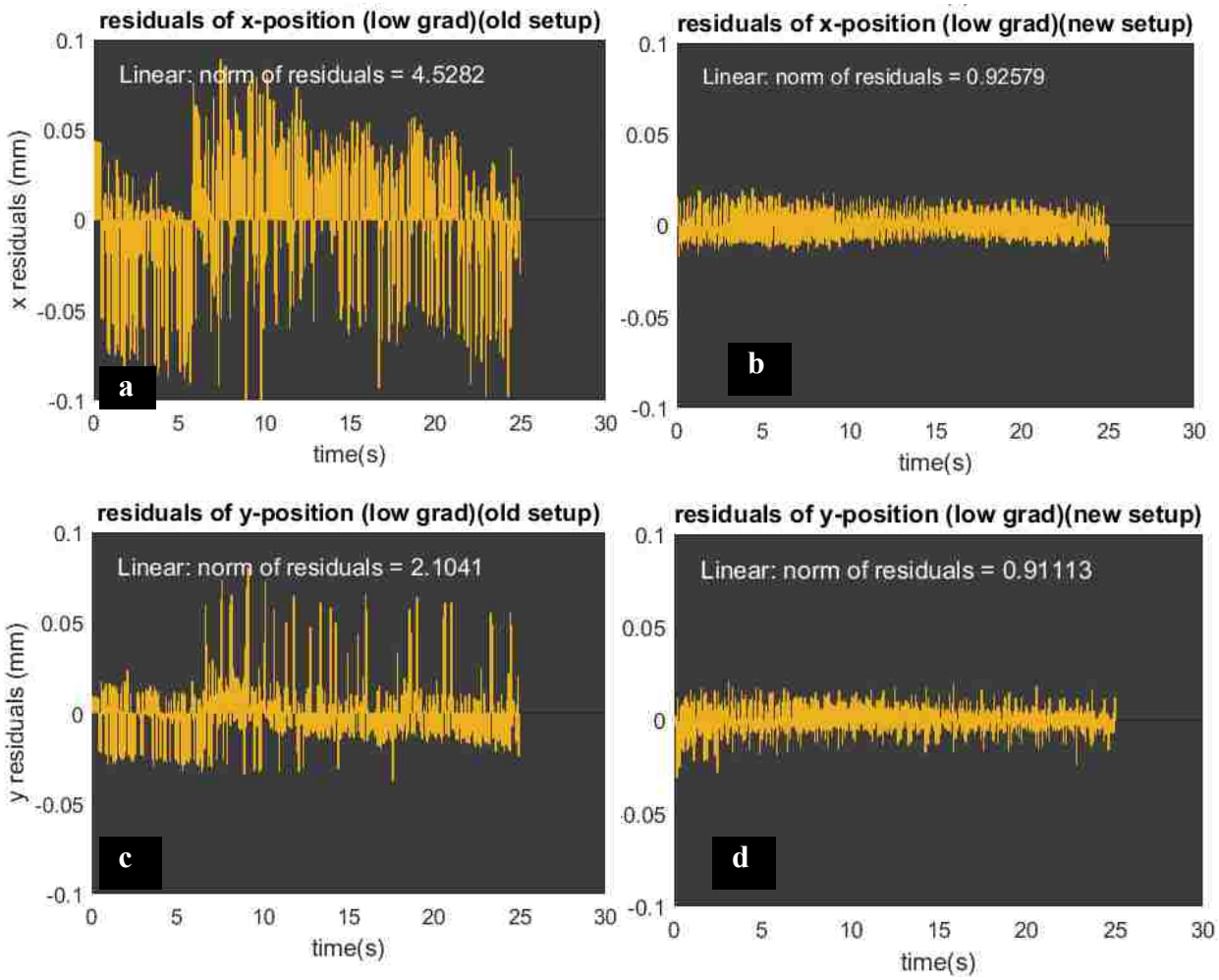


Figure 5.3 X and Y positions of recorded moving pattern with TI-weighted GRE sequence scan. a) X position of old design b) X position of new design over time c) Y position of old design over time d) Y position of new design over time

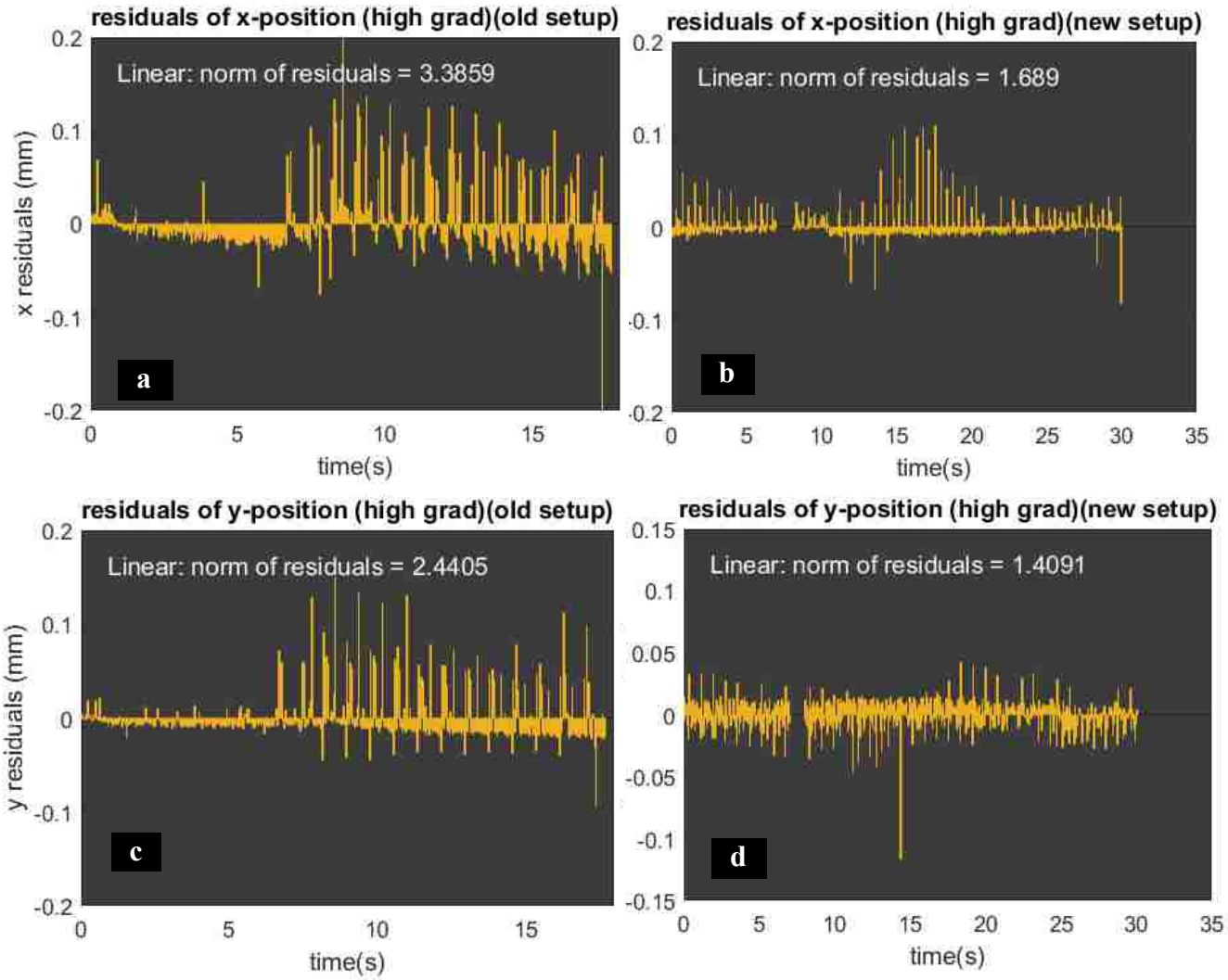


Figure 5.4 X and Y positions of recorded moving pattern with TI-weighted TSE QIR sequence scan. a) X position of old design b) X position of new design over time c) Y position of old design over time d) Y position of new design over time

The recorded positions for new design has less norms of residuals in all conditions, no gradient, low gradient, and high gradient. That means that the new design with 3D printed holder has less vibration regardless of the magnitude of gradient. The vibration is due to the moving of the holder, and that motion contributes to the noise of recorded patients' motion. There are drifts for both x and y direction; however, the new design has smaller drift change compared to the old design. There are several reasons that could cause the drift. Previous design relies on taping the

pipes with camera on the main coil, any loosen tape that does not hold all pipes in place can lead to small drift of the overall recording. The new design does not rely on the tape, but it has several joints that connect parts together, any small rotation in any of the joints will result in a drift in translational direction. Moreover, both the old and new designs hold the camera and laser on one side, and since the camera and laser is heavier than the pipe or the 3D holder with plastic bridge, the difference in weight between front and back sides can generate a small momentum that drags the design to rotate on one direction. Although the plastic bridge is securely embedded on the scanner's bed, and the drift is less than one pixel (< 0.127mm), this small rotation may lead to inaccuracy of classifying bulk motion (head turning) in real-time.

Previously, the retrospective motion correction method was tested on 3 volunteers. Among these volunteers, we found the smallest moving distance of the structured light that was recognized as abrupt motion was 0.01mm. Therefore, any vibration that is larger than 0.01mm will lead to a false positive result. The largest vibration of TSE QIR sequence, however, is 0.12mm, and the vibration under this sequence will cause a false positive result. This is a limitation of the mounting design, and this new mounting design could not be used for testing neck motion under sequence as violent as TSE QIR.

The assembly time of the new design is less than the old design, although we did not time the setup for every experiment, an approximate 20-minutes time reduction for the preparation work can be easily noticed. Furthermore, the new design is adaptable for all MR coils and free to move to any position along the bed.

The table below shows the norms of residuals, which describes the scale of vibrations, of the new mounting setup under 6 different gradients for the same GRE sequence.

	5 mT/m	10 mT/m	15 mT/m	20 mT/m	25 mT/m	30 mT/m
X norms of residuals	1.6099	1.4791	1.166	1.1911	1.25	1.1846
Y norms of residuals	3.21	3.1279	2.7698	2.6233	2.9907	2.4523

Table 5.1 Norms of residuals for x and y coordinates on 6 GRE sequences that only different in gradient.

We did not see an increase of vibration for higher gradients; however, a decrease trending in vibration can be noticed when the gradient is higher. We only tested the vibration under 6 gradients, and the increment of the gradient could be too small to affect the vibration. Therefore, these 6 scans could only represent the random vibration results under this specific GRE sequence. An increase of vibration can be seen on the 25 mT/m gradient, but this vibration is still less than 5 mT/m gradient. We would like to repeat this experiment several times with more gradients and on other sequences to determine the relationship between gradient and vibration. From these results shown here, we could not conclude the effect of the gradient alone on vibration.

Chapter 6. CONCLUSION AND FUTURE WORK

6.1 SUMMARY

We have successfully shown that changing from LabVIEW to Python for processing motion signal is more efficient and has less variance. The movement of the pattern that mimics the structured light on patient's neck can be accurately recognized and recorded by Python with

OpenCV library. The improvement of end-to-end latency is due to the stability and efficiency of the programming language.

We have also proved that the new 3D printed camera and laser holder improves the vibration problem of the old setup, and the calibration tests on vibrating noise show that the holder with plastic bridge on the scanner bed is steadier regardless of occurrence of scanning. A decrease in the assembly time can also be noticed with the new design. However, a false positive for abrupt motion due to the vibration will appear if TSE QIR sequence is used for scanning. Moreover, no relationship was found between gradient and vibration from the 6 scans we experimented.

6.2 Applications of the System

A possible application of our marker-less optical motion detection system is to replace the breathing belt. Breathing belt adopts a respiratory gating method that uses abdominal movement to find out the diaphragm end-expiratory position (DEEP) and applies a gating window that allows MRI machine to scan the patient only at DEEP [19]. This technique uses fiber optic cable to feed data into the scanner, and the noise level is lower since it has an external device attached onto the subject [20]. The respiration data acquired by our method matches well with the breathing belt technique, so we could potentially use our design with respiratory gating to eliminate artifacts caused by breaths, and our system is marker-less, which means it causes less inconvenience to patients. We could modify our optical data to fit in the windowing and use the previous embedded gating program in the scanner.

Another way to apply the marker-less optical motion detection is to communicate directly with the scanner in real-time. Ideally, we would like to pulse the scanner when abrupt motion (cough or swallow) occurs and resume the scanner when the motion ends, or we could re-scan the abrupt k-space lines while the patient is still inside the main coil. Both ideas require high accuracy of the

motion recognition and classification. Also, this ideal situation requires determining an efficient way of transmitting data from camera to the scanner. We currently use a BNC cable and USB adapter to communicate with the computer, but we would need to identify a more suitable way for communicating with the scanner.

6.3 FUTURE WORK

The next step of this project can be further decreasing the vibration of the holder by reducing the joints between parts. Also, a printer with higher resolution can be used to print parts that fit each other better. We could measure the vibration under more gradients or under different flip angles of same gradients to determine the factors for vibration. In this way, we would have a more quantitative understanding of the limitation for the new mounting design.

Besides that, feeding the processed motion signal into the scanner would significantly improve the use of the design presented in this thesis. We can either try to use the embedded gating method and imitate our processed signal as the diaphragm respiration data to apply on that technique, or we could try to control the scanner directly. Either way requires us to transfer data from BNC and USB cable into the scanner. The communicating between our program and the scanner will be a big, yet tough, step forward.

REFERENCES

1. Andre, J.B., et al., *Toward Quantifying the Prevalence, Severity, and Cost Associated With Patient Motion During Clinical MR Examinations.* J Am Coll Radiol, 2015. **12**(7): p. 689-95.
2. *Number of magnetic resonance imaging (MRI) units in selected countries as of 2015 (per million population)*. 2016 [cited 2017 May 25].
3. Gallagher, T.A., A.J. Nemeth, and L. Hacein-Bey, *An introduction to the Fourier transform: relationship to MRI.* AJR Am J Roentgenol, 2008. **190**(5): p. 1396-405.
4. Atkinson, D., et al., *Automatic compensation of motion artifacts in MRI.* Magn Reson Med, 1999. **41**(1): p. 163-70.
5. van der Kouwe A, B.T., Dale AM, *Real-time rigid body motion correction and shimming using cloverleaf navigators.* Magn Reson Med, 2006. **56**: p. 1019-1032.
6. Welch EB, M.A., Grimm RC, Ward HA, Jack CR, *Spherical navigator echos for full 3D rigid body motion measurement in MRI.* Magn Reson Med, 2002. **47**: p. 32-41.
7. Maclaren, J., et al., *Navigator accuracy requirements for prospective motion correction.* Magn Reson Med, 2010. **63**(1): p. 162-70.
8. Glover, G.H. and J.M. Pauly, *Projection reconstruction techniques for reduction of motion effects in MRI.* Magn Reson Med, 1992. **28**(2): p. 275-89.
9. Pipe, J.G., *Motion correction with PROPELLER MRI: application to head motion and free-breathing cardiac imaging.* Magn Reson Med, 1999. **42**(5): p. 963-9.
10. Tamhane, A.A. and K. Arfanakis, *Motion correction in periodically-rotated overlapping parallel lines with enhanced reconstruction (PROPELLER) and turboprop MRI.* Magn Reson Med, 2009. **62**(1): p. 174-82.

11. Taylor, A.M., et al., *Automated monitoring of diaphragm end-expiratory position for real-time navigator echo MR coronary angiography.* J Magn Reson Imaging, 1999. **9**(3): p. 395-401.

12. Liu J, C.H., Zhou Z, Wang J, and Yuan C, *Motion Detection and Correction Using Non-Marker-Attached Optical System during MRI Scanning.* Proc. Intl. Soc. Mag. Reson. Med., 2015. **23**.

13. *MR-compatible camera "12M" user manual*. 2015 May 22, 2015 [cited 2017 May 4]; 10:[Available from: http://www.mrc-systems.de/downloads/en/mri-compatible-cameras/manual_mrcam_12m.pdf.

14. Lunt, J., *Large-scale production, properties and commercial applications of polylactic acid polymers.* Polymer Degradation and Stability, 1998. **59**(1-3): p. 145-152.

15. Uecker, M.Z., S.; Voit, D.; Karaus, A.; Merboldt, K.; Frahm, J., *Real-time MRI at a resolution of 20 ms.* NMR Biomed, 2010. **23**(8): p. 986-994.

16. team, O. *About OpenCV*. 2017 [cited 2017 May 4]; Available from: http://opencv.org/about.html.

17. Gullichsen, S.S., *Delay in camera-to-display systems*, in *Computer Science*. 2011, University of Oslo: Oslo, Norway. p. 89.

18. Yuan., V.L.Y.a.C., *T1-Insensitive Flow Suppression Using Quadruple Inversion-Recovery.* Magn Reson Med, 2002. **48**: p. 899-906.

19. Ehman, R.L., et al., *Magnetic resonance imaging with respiratory gating: techniques and advantages.* AJR Am J Roentgenol, 1984. **143**(6): p. 1175-82.

20. Burdett, N.G., T.A. Carpenter, and L.D. Hall, *A Simple Device for Respiratory Gating for the Mri of Laboratory-Animals.* Magnetic Resonance Imaging, 1993. **11**(6): p. 897-901.

APPENDIX

```
import cv2
import time
import numpy as np
import argparse

y_dummy=0
detect=[]
#USB camera = 1; Default camera = 0
cap=cv2.VideoCapture(0)
cap.release()
cv2.destroyAllWindows()
cap = cv2.VideoCapture(1)

while(True):
    # Capture frame-by-frame
    ret, frame = cap.read()

    # Our operations on the frame come here
    gray = cv2.cvtColor(frame,cv2.COLOR_BGR2GRAY)
    ret, binary = cv2.threshold(gray,127,255,cv2.THRESH_BINARY)

    # construct the argument parse and parse the arguments
    ap = argparse.ArgumentParser()
    ap.add_argument("-i", "--frame", help = "path to the frame")
    args = vars(ap.parse_args())

    # load the image
    image = cv2.imread(args["frame"])

    lower = np.array([150,170,150])
    upper = np.array([255,255,255])
    shapeMask = cv2.inRange(frame, lower, upper)

    img2,contours, hierarchy = cv2.findContours(shapeMask,cv2.RETR_TREE,cv2.CHAIN_APPROX_SIMPLE)
    #complete contour
    cv2.drawContours(frame,contours,-1,(0,255,0),3)

    #rectanglular contour
    cnt = contours[-1]
    x,y,w,h = cv2.boundingRect(cnt)
    cv2.rectangle(frame,(x,y),(x+w,y+h),(0,0,255),2)

    M = cv2.moments(cnt)
    x = int(M['m10']/M['m00'])
    y = int(M['m01']/M['m00'])
    if abs(y-y_dummy)>2:
        ms=time.time()*1000
        detect.append(ms)
        print(y)
        y_dummy=y

    # Display the resulting frame
    cv2.imshow('frame',frame)
    time.sleep(0.900)
    if cv2.waitKey(1) & 0xFF == ord('q'):
        break

# When everything done, release the capture
cap.release()
cv2.destroyAllWindows()
print(detect)
```