Context-aware Automatic Video Screen Manipulation Using Trajectory Tracking

HyungJun Yoon
20217044

Seungjoo Lee
20160479

Abstract

Numerous people watch videos through monitor or smartphone screen. We point out the limitations of automatic screen manipulation according to the viewer's physical context when watching video through monitor. And to provide a better experience for video viewers, we present a service with context-aware 1) automatic screen size adjustment, 2) automatic screen rotation, and 3) automatic video switch between devices. We first show the improvement in trajectory tracking through hardware adjustment and integration error reduction, which leads to the improvement of the performance of our whole system. Then we describe the architecture of our system, consisting of integration of trajectory tracking, python client, web client, and mobile client. The working system is demonstrated through the linked video in the report.

1 Introduction

These days, a lot of people watch video through web-based platforms like Youtube or Netflix. Youtube have reported that the total number of hours people watch video in a day is 1 billion [6]. The main physical medium through which people watch these videos are monitors and screens of mobile devices. In order to allow users to have a better experience in watching a video, video screen interfaces have provided various functionalities within the mediums. Users can reduce or increase the size of the video, and can also rotate the video to a desired angle according to their preference.

Representative screen manipulation usages can be listed as follows. First, when users stay close to the screen, users may prefer a smaller screen for higher resolution and to display Overview Beside Details (OBD) [3], such as comments, on the screen together. On the other side, if the users get far from the screen, they usually prefer larger size since the content of the video in a small size is not enough to show the contents in a far distance. Second, when the direction of vision is changed, such as when people lie down, users expect the screen to be aligned with the direction of their vision. Finally, people don't watch video on only one device. In a situation where people move from place to place while watching videos on monitor, people mainly resume the same contents on their mobile phone, and this inter-device video switch also falls under screen manipulation.

Previously, these functions were usually set manually by active users. However, we assert that the aforementioned "user's preference" can be measured through sensors and that the screen can be passively manipulated through the measured context. With the insight, we propose a video screen manipulation application which automatically fits the screen according to the user context measured through an Arduino sensor. First, a sensor device is attached on a user's head. Since the required functionalities of the sensor are Bluetooth and IMU sensing, we expect that the sensor can be attached in the form of smart device, such as smart glasses, without inconvenience to users. The sensor measures the location and head orientation of the user in real time. The measured values are used for inferring the context of the user: how far the user is staying from the screen, the direction of the user's vision, and what device the user is focusing on. Based on the inferred context, our application decides on what device and in what form the video should be played. All communications between the sensor and screening devices are done by Bluetooth, and the video is manipulated through chrome and android API on monitor and phone, respectively. In the following sections we cover more detailed explanation.

2 Implementation

As shown in the Figure 1, our system is composed of clients that play video by communicating with sensors for context recognition and exchanging information in real time via Bluetooth. Context recognition through trajectory tracking is the most important part of our system. Since it is our big goal to precisely match the user's preference with the inferred context, we improve the performance of the sensor's location and orientation tracking through various technical improvements. Section 2.1 covers this in detail. Next, to manipulate

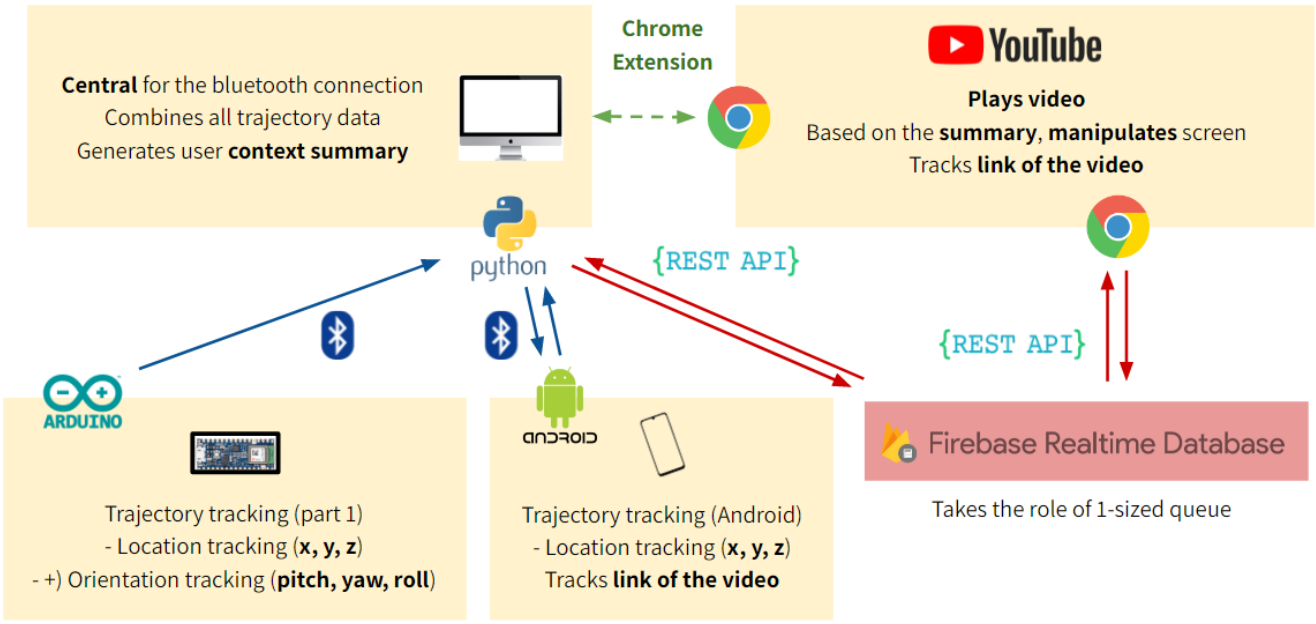


Figure 1: Service Overview

the screen of the actual video interface in the client, we implement the required functions in the web and app. We deal with front-end and back-end designs used in web and app clients, and communication between clients and sensors in section 2.2.

2.1 Trajectory tracking

In this subsection, we explain how we improved the trajectory tracking performance.

2.1.1 Hardware adjustment

- **Increasing sampling rate:** Since the default sampling rate of the Arduino's LMS9DS1 library [1] is too low (119Hz), we increased the sampling rate to 476Hz by changing the `CTRL_REG1_G` register [4] to capture the detailed movement.
- **FIFO mode:** If we use the most recent produced sensor value, we miss sensor values that are produced during calculation since the calculation for the trajectory tracking is usually slower than the sampling frequency. We enabled FIFO mode to save all produced sensor values to hardware queue by the library function `set_continuous_mode()` [1].
- **Average pooling from FIFO queue:** If we use one sample at a time, the FIFO queue will grow because of slow calculation. Then, we cannot achieve real-time trajectory tracking and we even lose some sensor values due to fixed FIFO queue size [4]. Thus, we revised the library function `read_acceleration()` to return the average value of all sensor values in FIFO queue.

2.1.2 Reducing integration error

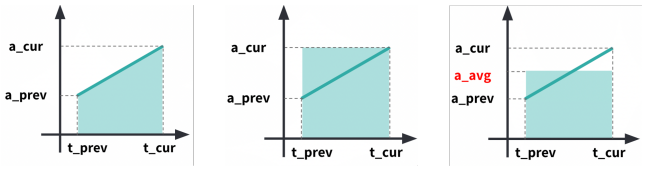
The baseline code integrates a velocity from a acceleration value using the most recent sensor value. However, this method introduces integration error. Figure 2 shows three integration methods. Figure 2a is correct integration when the sampling frequency is infinite. If we use current acceleration value like figure 2b, the result is bigger than the actual result.

Thus, we use the average of the previous and current acceleration values like figure 2c. This method can reduce the integration error with the assumption that the acceleration changes linearly in short time. Note that we also apply this integration scheme when integrates velocity to position.

2.1.3 Orientation tracking

To track the orientation, we integrate angular velocity from a gyroscope in LSM9DS1 [4]. For the gyroscope, We use the same hardware adjustments introduced in section 2.1.1.

Additionally, we calibrate the gyroscope beforehand. We measure the gyroscope value 200 times in steady state and average them. We subtract the average value when we use the gyroscope.



(a) Correct integration (b) Wrong integration (c) Adjusted integration

Figure 2: Three integration methods

2.1.4 Removing gravity in real-time

Our mobile service's usage scenario includes rotating and moving the device at the same time. To enable it, we should remove the gravity in real-time since the accelerometer cannot distinguish linear acceleration from gravity.

We calculate the rotated gravity in the arduino's coordination from rotation matrix R [5] and the calculated orientation (section 2.1.3) as shown equation below.

$$R^{-1}\begin{bmatrix}0 & 0 & 1\end{bmatrix}^T = \begin{bmatrix}-\cos(\alpha)\sin(\beta)\cos(\gamma)+\sin(\alpha)\sin(\gamma)\\ \sin(\alpha)\sin(\beta)\sin(\gamma)+\cos(\alpha)\sin(\gamma)\\ \cos(\beta)\cos(\gamma)\end{bmatrix}$$

2.2 Service

2.2.1 Python client

The Python client is placed within the device that plays a video through a web-based interface through a monitor. The Python client has two main roles. First, as a central device of Bluetooth, the client receives data from the smartphone and Arduino sensor through Bluetooth. It receives trajectory data from the sensor and the status of the video being played from the mobile client. Next, it serves to create a summary of the user context based on the transmitted data. The generated summary is delivered to a database available to web clients and directly to mobile clients.

2.2.2 Web client

The web client executes a script that directly manipulates the video based on the generated context summary. This is implemented through a chrome extension script that manipulates the video player in YouTube in various ways through javascript injection. The Chrome extension cannot communicate directly with the Python script, so it uses Firebase Realtime Databse in the form of a 1-sized queue to deliver data to the Python client through REST API.

2.2.3 Mobile client

The mobile client executes a script that plays a video through a WebView within the Android app. Information related to the state of the video is transmitted to the Python client through Bluetooth along with sensor data. Whether or not the video is played is judged according to the context summary received from the Python client. This is for real-time device switching according to the user's focus.

(a) Distance tracking evaluation setting (b) Orientation tracking evaluation setting

Figure 3: Evaluation settings

3 Evaluation

We evaluate the trajectory tracking and mobile service at the same time by measuring the performance of distance tracking, orientation tracking, and gravity cancellation. We believe that it is sufficient to evaluate the mobile service with these three evaluations because the mobile service solely depends on the trajectory tracking performance.

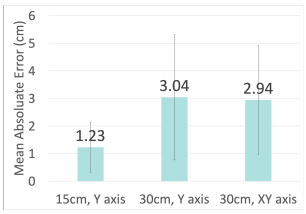
3.1 Distance tracking

We evaluate the distance tracking by moving the arduino in straight line as shown in figure 3a. We moved the arduino to two different distances (15cm, 30cm). We also tested with two different axes (Y axis, XY axis) to test robustness of the distance tracking. We tried 20 times for each combination of distance and axis, and report mean absoluate error with standard deviation.

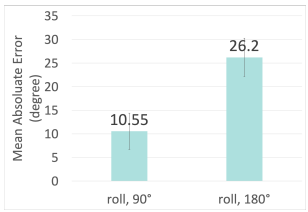
Figure 4a shows the performance in mean absolute error. The error was about 10% to the total distance. When the distance increases, the error increases as well. We also confirmed that the error is same for different axes.

3.2 Orientation tracking

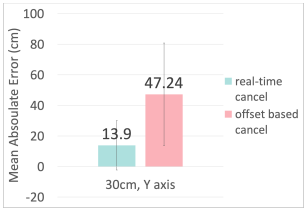
Orientation tracking performance is evaluated by rotating the arduino and changing roll as shown in figure 3b. We tried changing roll 90 and 180 degrees, 20 times each. We evaluated only roll without loss of generality since roll, yaw, pitch are calculated in the same way.



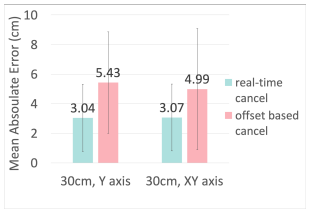
(a) Distance tracking



(b) Orientation tracking



(c) Gravity cancellation with rotation



(d) Gravity cancellation in straight line

Figure 4: Evaluation results

Figure 4b shows the mean absolute error in degree. The error was about 10° when we tested for 90°. The error increases when we tested for larger degree.

3.3 Gravity cancellation

We evaluate the gravity cancellation performance by combining rotation and distance tracking. In specific, we moved the arduino to following 3 steps for 20 times. 1) Rotate the roll +90°, 2) Rotate the roll -90°, and 3) Move 30cm to Y axis. We also evaluated moving the arduino just in straight line 30cm to 2 different axes (X, XY axis) for 20 times.

Figure 4c shows mean absolute error of rotation and moving. The pink one is the baseline performance that cancels the gravity with offset-based method. The skyblue one cancels the gravity in real-time with the proposed method. We can confirm that our method effectively cnacels the gravity in real-time and reduces error greatly.

Figure 4d shows the mean absolute error of moving the arduino in straight line. It can be seen that even when moving in a straight line, the error is reduced with our method. This is because the hand is shaking slightly and changes the gravity in the arduino's coordination.

3.4 Application feasibility test & Demo

Totally, our implemented system involves seven basic functionalities for automatic context-aware screen manipulation.

- Scaling up when the user gets far from the monitor than the threshold
- Scaling down when the user gets close to the monitor than the threshold
- Rotating the screen to left when the user's head rotates to left
- Rotating the screen to right when the user's head rotates to right
- Rotating the screen to the original direction as the user's head gets to the original rotation.
- Switching the playing device to smartphone when user's head is directing smartphone
- Switching the playing device to monitor when user's head is directing monitor

To verify that our system works with all of the listed functionalities, we conducted a simple feasibility test, by performing all situations corresponding to the functionalities. Our conducted experiment is shown in the demo video(https://youtu.be/a_yV5p-jiIo). As a result, our system worked for all scenarios with the expected manipulation functionality.

4 Discussion & Challenge

- **Fixation of smartphone location:** We fixed the smartphone location due to great location tracking error. Since the android's sensor value is gathered by eventlistener, it was hard to get the sensor value in desired sampling frequency. We believe that we can track the smarpthone's location with RF-based methods [7], and not fix the location.

- **Using arduino as peripheral device:** Currently, the arduino and smartphone are set as peripheral device and the laptop is set as central device. We should set the arduino as central device to enable more diverse usage scenario, such as taking the phone to outside. Making the laptop peripheral was hard due to limitation of Bleak library [2]. We believe this limitation can be solved with other libraries that enable the laptop playing the peripheral role.

Roles

- Part 1. trajectory tracking
  - HJ : reducing algorithmic error (integration error)
  - SJ : hardware adjustment, gravity cancelling, orientation tracking
- Part 2. mobile service
  - HJ : implementing summary generation & system architecture, youtube screen manipulation functionality (chrome extension)
  - SJ : bluetooth connection & data transfer among devices (arduino, phone, PC)

Acknowledgments

We appreciate the supervision of Professor Song Min Kim who provided knowledge about sensors and wireless communication through high-quality classes, and guided us to proceed with valuable projects.

References

[1] Arduino. Arduino lsm9ds1 library. Retrieved December 17, 2021 from https://www.arduino.cc/en/Reference/ArduinoLSM9DS1.

[2] Bleak. Bleak library. Retrieved December 19, 2021 from https://bleak.readthedocs.io/en/latest/.

[3] Kasper Hornbæk and Erik Frøkjær. Reading of electronic documents: The usability of linear, fisheye, and overview+detail interfaces. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '01, page 293–300, New York, NY, USA, 2001. Association for Computing Machinery.

[4] STMicroelectronics. Lsm9ds1 datasheet. Retrieved December 17, 2021 from https://www.st.com/resource/en/datasheet/lsm9ds1.pdf.

[5] Wikepedia. Rotation matrix. Retrieved December 18, 2021 from https://en.wikipedia.org/wiki/Rotation_matrix.

[6] Youtube. Everyday people watch 1 billion hours of videos on youtube and generate billions of views. https://www.youtube.com/intl/en-GB/about/press/, 2019.

[7] Faheem Zafari, Athanasios Gkelias, and Kin K. Leung. A survey of indoor localization systems and technologies. *IEEE Communications Surveys Tutorials*, 21(3):2568–2599, 2019.