Mid-Semester Thesis Report: Submitted in partial fulfillment of the requirements of BITS 422T Thesis

by

© *Nehil Jain* (ID - 2008B4A3560G)

Under Supervision of *Prof. Jeremy R. Cooperstock*

Department of *Electronics and Electrical Engineering*

BITS, PILANI - K.K.BIRLA CAMPUS

May 2013

Nehil Jain McGill University

Abstract

In this report I focus on two aspects of my research projects. First will be the techniques used to enhance Walking Straight Application using Computer Vision. Later, I shall describe the work on Real-Time Emergency Response: Modern Tools for Modern Disasters.

"The purpose of the abstract, which should not exceed 150 words for a Masters' thesis or 350 words for a Doctoral thesis, is to provide sufficient information to allow potential readers to decide on relevance of the thesis. Abstracts listed in Dissertation Abstracts International or Masters' Abstracts International should contain appropriate key words and phrases designed to assist electronic searches."

— MUN School of Graduate Studies

Acknowledgements

Put your acknowledgements here...

"Intellectual and practical assistance, advice, encouragement and sources of monetary support should be acknowledged. It is appropriate to acknowledge the prior publication of any material included in the thesis either in this section or in the introductory chapter of the thesis."

— MUN School of Graduate Studies

Contents

List of Tables

List of Figures

Chapter 1

Walking straight

1.1 Introduction

A well-known problem associated with walking without vision is veering, which is the inability to maintain a straight path. Many studies conducted previously have proven the veering effect and its consequences. The reason is uncertain but is hypothesized to be motor error in stepping movement. Blind people face the same problem while walking. This makes crossing intersections and walking on a straight path dangerous and unsafe.

Walking Straight an application for the smartphone device was built by Paneels et al. [3]. It uses the available hardware on a smartphone device, the iPhone 4, to provide real-time audio feedback to the blind users in order to correct their deviation. The application is built using the ISAS (In-Situ Audio Services) [1] system architecture. The application uses compass and gyroscope sensors to calculate the deviation from a straight path. The values from the sensors are filtered to remove body sway of a person

while walking. Different auditory feedback designs were evaluated experimentally and their performance was evaluated. A continuous tone played in the ear in the side of the deviation is concluded to be the most efficient way of rendering the audio feedback to the blind [3].

1.2 Are the sensors reliable?

The study of todays smartphone sensors reliability was done by Blum et al.[4]. The current state of the walking straight algorithm assumes a stable 10Hz Gyroscope. It solely depends on the sensors to calculate the heading.

For example, MUN's logo can be seen in Figure 2.1.

Figure 1.1: This is MUN's logo

Figure XX indicates compass values (cyan) against ground truth (black), the latter of which is constant (horizontal) for each straight-line leg of the walk. Actual compass error (red) is calculated as the absolute difference between these two, whereas reported compass error (grey) is an estimate of error magnitude by the sensor itself. As can be seen, the actual error fluctuates both above and below the estimate. Gaps in the plot represent transitions between legs of the walk, during which we have no ground truth heading information.

Yaw (green), obtained from the gyroscope sensor, is not calibrated to north, so this only represents relative variation. This data is should be a flat line, excepting body sway while walking. Slope in yaw indicates drift, observed in all legs of this walk. The reported course (purple) is derived from the direction of travel based on previous location updates. Both the iPhone and Android devices report course and speed based on location changes, but these appear to be of limited use even in the constrained straight-line testing they performed [4]. The inaccurate results of this study indicate that the application cannot solely rely on these sensors.

1.3 Computer vision solution

I have been working on overcoming the sensor inaccuracies, using computer vision techniques on the video captured from a smartphone camera, to generate a better feedback for the blind than what is present currently.

1.3.1 Problem

Two kinds of feedback are required by blind people to cross the intersection: when to cross and audio feedback to maintain a straight path. The correct time to cross the street can be determined by the state of the traffic light. The traffic light recognition can be done using color segmentation, shape segmentation or template matching or a combination of these.

1.3.2 Colour Segmentation

Color Segmentation is the technique to partition an image into different sets of pixels. This can be done using the specific color of the object we want to identify in the frame. I used OpenCV 2.4 to achieve this. I calculated the Hue, Saturation, and Value (HSV). The image from the source, in RGB format, was converted to HSV and then standard thresholding technique was applied. HSV is a color model based on human vision. The image(right) in Fig. XX is the binary image, the red cup being shown with white pixels.

1.3.2.1 Why HSV rather than RGB?

- The simple answer is that unlike RGB, HSV separates Luma, or the image intensity, from Chroma or the colour information. This is very useful in many applications. For example, if you want to do histogram equalization of a colour image, you probably want to do that only on the intensity component, and leave the colour components alone. Otherwise you will get very strange colours.

- In computer vision you often want to separate colour components from intensity for various reasons, such as robustness to lighting changes, or removing shadows.

Using thresholding only, we can track the traffic light, but that will not be accurate for all cases. There might be a few false detections.

1.4 False Detections

For a relatively simple case, such as a sky background, the color-based recognition method can effectively detect and identify traffic light. For a relatively complex situation, such as an urban environment, false detections will appear easily using the color-based recognition method. The shape-based feature recognition method can effectively reduce the false detections of the color-based feature recognition. But, the different shape characteristic rule has to be created for the different styles of traffic lights. This limits the flexibility of the algorithm. For the recognition method based on the template matching, the different styles of traffic light templates also have to be created to realize the recognition of the different style of the traffic light.

1.4.1 Labelling

To overcome this problem of false detections in the scene, labeling the candidate region of the traffic light is one of the most feasible solutions. I intend to use the Canny Edge Detector, that is inbuilt in OpenCV libary, which makes it lightweight and efficient. We can detect the closest bounding rectangle of the light and then label the red and green lights accordingly. In this way we can eliminate any other object, which was previously identified as a potential traffic light. This method was proposed by Gong et al. [2].

1.5 Tracking

I plan to generate the feedback using OpenCVs capabilities to extract motion from a video sequence using optical flow. Optical flow assess motion between two frames without any prior knowledge about the content of the frames. There are two types of tracking techniques, Dense Tracking technique and Sparse Tracking technique. Dense tracking technique utilizes each pixel in the frame to create a motion vector. This is computationally heavy and seldom used nowadays. We will use Sparse Tracking algorithm which calculates the motion vector for a subset of pixels

1.5.1 CAMShift

CAMShift is almost ideal for traffic light tracking from a pedestrians perspective for multiple reasons. One, the speed of the camera (pedestrian) is low. The traffic light (object) never disappears from the frame. Camshift allows for tracking objects whose size may change during the sequence.

1.6 Future Work

The larger goal of the algorithm I am building is to help blind people walk straight whenever they wish to, irrespective of what is in the scene (e.g. Intersection). Feature extraction from the scene can be used to generate accurate feedback for the blind user. Corner and edge are the features will be suitable for our specific problem. I will implement SIFT and FAST corner detector algorithm for the edge detection. These algorithms are Scale and illumination Invariant. FAST has been tested on iPhone

and is quite efficient.

Chapter 2

Real time Emergency Response(rtER)

2.1 Introduction

I will describe our Real Time Emergency Response (rtER) prototype system, designed to help manage real-time information during a crisis, largely focused on identifying and filtering the most critical information. The real-time Emergency Response project deals with the detection, observation, and assessment of situations requiring intervention by emergency responders. It offers them access to high-quality live" data that may be visualized effectively both by responders in-situ and by remote operators in dedicated control rooms. Unlike other systems that focus on still images and textual content, rtER also incorporates live video feeds. It also explores the utility of establishing a two way link between a person shooting a video from a smartphone camera and the other person directing the view of the camera from a remote location,

to guide them to a better vantage point, implemented seamlessly into the web-based user interface. The system has following components

1. Smartphone application.
2. Web server component.
3. Client web application.
4. An immersive visualization environment.

In emergency response and crisis management, the flow of information is tightly controlled through institutionalized hierarchical channels. Public perception can be key in preventing mass panic and maintaining public cooperation. The users of the different strata of group activity are:

- First Responders or emergency response team (ERT): need filtered information or results as immediate action needs to be taken
- Bystanders: provide real-time information and reports, may be unreliable
- Public Information Officer (PIO), typically situated at the emergency operations center (EOC), where coordination and dispatch are handled
- Virtual volunteers, arranged in different trust levels, and work closely with the PIO
- The general public

The use of the different technologies developed, serve different user groups as illustrated in figure XX

2.2 Problem

The following scenario (based on real-life events) helps understand the motivation for building rtER.

> *" A 48-inch water main broke at the top of the hill that descends into downtown Montreal, right through the main McGill campus. There was massive flooding through all of the streets and into many of the buildings on the surrounding campus, with black ice underneath, making it practically impassable in many spots. The campus was almost completely surrounded by water making it challenging to find a way off campus.*
>
> *Jeff left just after the main break, but before any official news had been sent. He could not get off campus easily since the water was flowing too deep and fast through the streets, and had no way of figuring out how far the situation spread without slogging through knee-deep water. There were many students and others standing around with cameras taking pictures and video all around campus (later dozens of videos were posted online) and posting on social media.*
>
> *For those in the buildings trying to plan their way home, being able to see these images, videos and tweets might have been helpful. But without some way of sorting and filtering it might also simply have been overwhelming.*
>
> *We imagine that rtER would have helped us solve this problem. By allowing information—video, image or text—to be visualised in geographic context on the rtER map, users could have quickly understood what was occurring where. Moreover, filtering using the map would have provided a way for those looking for a way of campus to eliminate all superfluous information and see only the information coming from one area on their map. This would mean you could quickly confirm or deny if a route was safe.*
>
> *Since we are trying to build our system with collaboration in mind, repetitious search would be avoided and time could be saved. For example in this case most people posting pictures and videos were "ambulance chasing" by showing the drama and devastation, not the less interesting safe area. This means that the "useful" information was being buried by the "shock value" videos. Once a person had found a safe passage using our system, via a tweet suggesting a safe route or possibly a picture or a video showing an unflooded area, they could promote that information to higher priority for others to see. As a result the next person wouldn't need to*

repeat that search. They might instead simply add a confirmation after using the route, or perhaps provide an alternative. In this way we hope our system may make a more efficient use of human resources.[5-rter big data paper] " [1]

2.3 System Architecture

I was involved in the development of two components of the whole system, viz., android client application and the client web application.

2.3.1 Collaborative information web client

The web client was developed for the virtual operations support teams(give reference)(VOSTs). With the increasing public participation in emergency response, the number of incoming videos and social media streams can easily overwhelm the EOC personnel. The tool developed for this group of users, built around a web client (Figure XX), can be used by emergency responders to curate information including live video, Twitter feeds, YouTube and other social media. My worked was focused on Twitter integration in the Web Client.

2.3.1.1 Importance of Twitter in crisis management

The different stages of any disaster are:

1. Warning
2. Threat
3. Impact

4. Inventory

5. Rescue

6. Remedy

7. Recovery

Micro-blogging has been proven to be highly effective as one of the useful tools in stages 3 to 5 according to the above taxonomy. During these stages, more traditional communication channels are less effective than the emerging ICT ones. Twitter tends to describe a common/global topic, diminishing the network entropy. This makes Twitter an indispensable tool.

The backbone of the web client is HTML5 and AngularJS. AngularJS is an open-source JavaScript framework, maintained by Google. The framework adapts and extends traditional HTML to better serve dynamic content through two-way data-binding that allows for the automatic synchronization of models and views. This make our web client app responsive and provides near real-time updates in a particular web view.

The UI is built using AngularUI and Twitter Bootstrap UI. AngularUI is the companion suite(s) to the AngularJS framework. It comprises of several components: UI-Module, UI-Bootstrap, NG-Grid, UI-Router and IDE plug-ins. The Twitter Bootstrap UI is a front-end framework for faster and more elegant web development. This framework comprises of responsive css. This makes our web client app have a uniform UI on latest desktop, tablets and smartphone browsers as well.

The present implementation of the web client has two kinds of Twitter Items.

- Twitter search : This is a stream of relevant tweets that match a specified query. This feature uses the Twitter Search API. Get Search API v1 resource url : `http://search.twitter.com/search.format`. There are three views associated with each Twitter search item in the system, viz., Creation, Close-Up View and Tile View.
  - Create View: The form to create a search query is shown below. The user can build their specific query using 3 options.
    * Search Term: The default functionality is to search the term entered in the field as a keyword in the tweet. More complex queries can be constructed using operators. Example Query: `"This exact phrase" any OR of OR these OR words -none -of -these -words #these OR #hashtags lang:en from:from OR from:these OR from:accounts to:to OR to:these OR to:accounts mentioning OR these OR accounts near:"location" within:15mi :(`. Operators explained
      · Words within `" "` - This exact phrase
      · Words separated by `OR` - Any of these words
      · Words preceded by `-` - None of these words
      · Words preceded by `hboxes/vboxes`
      · `lang:` - language selection
      · Words separated by `from:` - from these user accounts
      · Words separated by `to:` - to these user accounts
      · Words separated by - mention of these user accounts

· **near:** - location selection

· `:),:( , ?` - Positive sentiment, Negative sentiment, Question, respectively.

* The user can also select the result type of the stream from the following options:

· `mixed` - Include both popular and real time results in the response.

· `recent` (Default) - Return only the most recent results in the response.

· `popular` - Return only the most popular results in the response.

* The user can make the search query, location specific also. Using the map view, the user can draw a circular region on the map to define the area of search.

– Close Up View : This view displays the stream of tweets resulting from the query. Each tweet can be promoted to the common live feed as a Single Tweet. Also the user can react to the tweet instantly.

– Tile View: This view represents the twitter search item as a part of the live feed in the form of an item of a grid of tiles. This emphasizes on the search term.

• Single Tweet: It is a single tweet that is promoted by the VOST members to promote it into the collaborative workspace.

– Create View: The Single tweet can be created by promoting a tweet from twitter search stream to the mainstream.

– Close up Item: The Close-up view displays the tweet card of the particular tweet. Tweet card displays tweet with expanded media like photos, videos, and article summaries, and also include real-time retweet and favourite counts.They are interactive and enable the users to follow the Tweet author, and reply, re-tweet, favourite all directly from the page. To achieve this, I used the oEmbed endpoint of Twitter. Resource url: `https://api.twitter.com/1/statuses/oembed.format`

– Tile Item: This view represents the twitter search item as a part of the live feed in the form of an item of a grid of tiles. This displays the text of the tweet in the tile to get a quick reference to the Single tweet.

2.4 Video Streaming

Previously, our system architecture lacked a robust video streaming architecture. Our latest video streaming architecture, shown in Figure 7, provides end-to-end scalable and relatively low-latency video from a smartphone source to HTML5 web clients, supporting ingest of multiple simultaneous streams at a server and scalable distribution to multiple simultaneous viewers.We achieve these goals by employing live encoding at the smartphone, transcoding streams into segmented video at a server and delivering video segments to web browsers using HTTP Live Streaming (HLS) [10] in two different formats, MPEG2-TS/H.264 and WebM/VP8. On the ingest side, our smartphone application encodes captured video frames into H.264/AVC [11], encapsulates them into a timestamped MPEG2 Transport Stream and forwards the video frame-wise via HTTP to our streaming server. I am involved in developing the

Android client application.

2.5 Android Client App

The typical use case for the mobile app places someone with a smartphone on site in a disaster scenario, ideally situated to provide relevant contextual data to emergency responders, possibly filtered by virtual volunteers. Using the rtER mobile app, the smartphone user streams a video feed, deemed to be relevant, which immediately appears as a live tile to VOST volunteers connected to the rtER web client, where it can be sorted based on its situational importance.

- Version 1 - For the live video streaming, we used the open source ipcamer-areference required, a project for android, along with `Google LibJingle` and `Nanohttpd`. These provide a peer to peer data transfer from the android device, acting as a server on a wifi network. Timestamped latitude and longitude information obtained using location-based services, are delivered to the server, in parallel to the live stream as a JSON object. This implementation creates a server on the android client which is blocked by the network providers firewall. Thus, the live video data cannot be sent.

- Version 2 - In this version, we developed a new app from ground up. It streams jpeg images from the camera and provides a feature for directing the viewer to look at a desired vantage point. This created a two way link between the curator and the user of the app capturing the stream. The images are captured using Camera Class in Android SDK and the desired orientation of the device

is indicated by the bounding box created in OpenGL, overlaid on the preview surface of the camera. This is illustrated in the figures below. The frame rate is low because each frame is captured after the previous has been written to the SD card(external storage) of the device. Then it is added into the queue, to be posted to the server.

- Version 3 -Currently, I am working on improving the frame rate to make the stream almost real-time video over 3g network. There are two possible ways to achieve this on android platform:

  - `MediaRecorder` class: This is a class in android SDK, used to record audio and video from the device. The recording control is based on a simple state machine. H.264 encoded video is written to a Mpeg2 TS file. Android SDK Media API is one of the weak points of this platform. The API restricts the developer from writing the recorded data from the camera to a stream. Presently, this data can only be written to a file in the external/internal filesystem of the device. To circumvent this, I plan to write the file to a local socket using java.net.Socket. Network sockets also have file descriptors and they can be accessed using, `ParcelFileDescriptor`. Using this, we can plugin the video stream data into the socket. On the other end of the socket, we can retrieve the data and use HTTP Post to sent to the video server using `ChunkedOutputStream` in android. The advantage of this method is, it uses the devices hardware encoders. This will be faster and computationally inexpensive.

  - FMpeg and Libx264: This method involves extracting raw frames from the

camera and encoding them to H.264 using FFMpeg Library. This can be achieved using Android Native Development Kit(NDK). Then H.264 encoded frames can be individually sent to the server via HTTP Post. The server handles the packaging of these frames into a Mpeg2TS stream. This method is comparatively slower and less efficient, as it uses more resources. Our iOS implementation uses the same architecture. The optimum parameters of the video stream transmitted is 15 frames per second(fps) at a resolution of 640 x 360 at 600kbps using iPhone 4s devices. This is susceptible to network congestion.

This is a work in progress.

We can include encapsulated PostScript™ figures (`.eps`) in the document and refer to it using a label. For example, MUN's logo can be seen in Figure 2.1.

Figure 2.1: This is MUN's logo

Figure 2.2 shows a chart of MUN's Fall enrollment from 2005 – 2009.[1] The figure

[1]From *Memorial University of Newfoundland — Fact Book 2009.*

Figure 2.2: MUN Fall Enrollment 2005 – 2009

was created using the Calc spreadsheet application of the office suite OpenOffice.org.[2] This figure was reduced by 50%.

For larger figures, we can use landscape mode to rotate the page and display the figure using the `\munlepsfig` command, as shown in Figure 2.3. The figure will be the only thing on the page when typeset in landscape mode. (The figure is reduced to 85% of its original size.)

Alternatively, if we just want to rotate the figure, but not the entire page, we can specify an `angle` attribute in the default argument of the `\munepsfig` command. The result is shown in Figure 2.4. If the figure is too large or if there isn't sufficient text, then the figure may appear on its own page.

Note that all three of the enrollment figures are basically the same file, but with different names — on Linux, they are symbolic links to the same file. The filenames

[2]This office suite can be downloaded at no cost from `http://openoffice.org/`. Unlike other commercial office suites, OpenOffice.org may be legally shared with colleagues and fellow students. There are versions for Linux, Microsoft Windows, Mac OS X and Solaris. Also, unlike commercial offerings, OpenOffice.org does not require activation using registration keys.

Figure 2.4: MUN Fall Enrollment 2005 – 2009 (rotated)

have to be different because the reference labels need to be unique.

Figure 2.5 shows a Petri net created using the `xfig` program (`http://www.xfig.org/`) which has very good support for LaTeX. This figure has been reduced to 40% of its original size.

Figure 2.5: A deadlocked Petri net

We can also create figures of text (such as short code snippets) using the `\muntxtfig` command, as show in Figure 2.6.

```
#include <stdio.h>

int main(int argc, char **argv)
{
  printf("Hello world!\n");
  exit(0);
}
```

Figure 2.6: Hello World

2.6 Tables

We can also create tables, as seen by Table 2.1. Note that, as required by SGS guidelines, the caption for a table appears above the table whereas figure captions appear below the figures. Tables and figures can "float" — they may not appear on the page on which they are mentioned. LaTeX tries to handle figure and table placement intelligently, but if if you have a lot of them without a reasonable amount of surrounding textual content, the figures and tables can accumulate towards the end of the chapter. Generally speaking, if there is sufficient text explaining the tables and figures or if the tables/figures are relatively small, this may not be a problem. However, if you have a lot of tables or figures, it may be a good idea to put them in an appendix and refer to them as the need arises.

Table 2.2 shows a different table in landscape mode.[3] This is useful if your table

[3]This data was also taken from the *Memorial University of Newfoundland — Fact Book 2009*.

Table 2.1: Fall Semester Enrollment

	Undergraduate			Graduate		
	F/T	P/T	Total	F/T	P/T	Total
2004	13,191	2,223	15,414	1,308	879	2,187
2005	13,184	2,143	15,327	1,375	920	2,295
2006	12,809	2,224	15,033	1,373	899	2,272
2007	12,634	2,155	14,789	1,403	899	2,302
2008	12,269	2,208	14,477	1,410	1,005	2,415
2009	12,382	2,323	14,705	1,567	1,106	2,673

is too wide for the page. Tables are double-spaced by default. To single-space a table, change the `\baselinestretch` before beginning the table environment. Remember to restore it after the environment has ended.

Table 2.2: Masters Degrees Conferre

	2009		2
	May	Oct	Ma
Degrees			
Master of Applied Science	14	2	1
Master of Applied Social Psychology	1	5	
Master of Applied Statistics	0	0	
Master of Arts	37	49	2
Master of Business Administration	14	16	2
Master of Education	107	87	12
Master of Employment Relations	8	9	
Master of Engineering	20	19	2
Master of Environmental Science	3	3	
Master of Marine Studies	2	0	
Master of Music	4	1	
Master of Nursing	7	8	1
Master of Oil and Gas Studies	0	0	
Master of Philosophy	5	4	
Master of Physical Education	0	2	
Master of Public Health	0	8	

Bibliography

[1] L. Lamport. *LaTeX: A Document Preparation System*. Addison-Wesley Publishing Company, second edition, 1994.

Appendix A

Appendix title

This is Appendix A.

You can have additional appendices too (*e.g.*, `apdxb.tex`, `apdxc.tex`, *etc.*). If you don't need any appendices, delete the appendix related lines from `thesis.tex` and the file names from `Makefile`.