SPATIAL SENSOR DATA PROCESSING AND ANALYSIS FOR MOBILE MEDIA APPLICATIONS

WANG Guanfeng

(B.E., ZJU, CHINA)

A THESIS SUBMITTED

FOR THE DEGREE OF DOCTOR OF PHILOSOPHY
SCHOOL OF COMPUTING
NATIONAL UNIVERSITY OF SINGAPORE

2015

DECLARATION

I hereby declare that this thesis is my original work and it has been written by me in its entirety. I have duly acknowledged all the sources of information which have been used in the thesis.

This thesis has also not been submitted for any degree in any university previously.

WANG Guanfeng Jan 20, 2015

ACKNOWLEDGEMENTS

This thesis is a summary of my four years research work. I am deeply grateful to the school for its support throughout my whole Ph.D. programme and more importantly, the wonderful research resources and brilliant people here successfully equipped me with the knowledge and skills that made this work possible.

I owe a double debt of gratitude to my supervisor, Roger Zimmermann. He guided me each step of the way on how to do research and how to become an eligible researcher. His advices on my work, commitment to academics and care for students are always my source of inspiration and encouragement whenever the difficulties seemed overwhelming.

I have also benefited greatly from the discussions and collaborations with my colleagues. My sincere thanks go to Beomjoo Seo, Hao Jia, Shen Zhijie, Ma He, Zhang Ying, Ma Haiyang, Fang Shunkai, Zhang Lingyan, Wang Xiangyu, Xiang Xiaohong, Xiang Yangyang, Gan Tian, Yin Yifang, Cui Weiwei, Seon Ho Kim, and Lu Ying from both NUS and USC.

I would also like to thank my flatmates, with whom I spent most of my spare time in Singapore. We had great moments together and these cheerful and precious memories will never fade away.

I dedicate this thesis to my parents and all my beloved friends. As an East Asian, it is not always easy to express my feelings in words, but I know for sure that I love them and I am forever grateful for their timeless love and unconditional support.

CONTENTS

SUMMARY

Currently, an increasing number of user-generated videos (UGVs) are collected and uploaded to the Web – a trend that is driven by the ubiquitous availability of smartphones and the advances in their camera technology. Additionally, with these sensor-equipped mobile devices, various spatial sensor data (*e.g.*, data from GPS, digital compass, etc.) can be continuously acquired in conjunction with any captured video stream without any difficulty. Thus, it has become easy to record and fuse various contextual metadata with UGVs, such as the location and orientation of a camera. This has led to the emergence of large repositories of media contents that are automatically geo-tagged at the fine granularity of frames. Moreover, the collected spatial sensor information becomes a useful and powerful contextual feature to facilitate multimedia analysis and management in diverse media applications. Most sensor information collected from mobile devices, however, is not highly accurate due to two main reasons: (a) the varying surrounding environmental conditions during data acquisition, and (b) the use of low-cost, consumer-grade sensors in current mobile devices. To obtain the best performance from systems that utilize sensor data as important contextual information, highly accurate sensor data input is desirable and therefore sensor data correction algorithms and systems would be extremely useful.

In this dissertation we aim to enhance the accuracy of such noisy sensor data generated by smartphones during video recording, and utilize this emerging contextual information in media applications. For location sensor data refinements, we take two scenarios into consideration, pedestrian-attached sensors and vehicle-attached sensors. We propose two algorithms based on Kalman filtering and weighted linear least square regression for the pure location measure-

ments, respectively. By leveraging the road network information from GIS (Geographic Information System), we also explore and improve the map-matching algorithm in our location data processing. For orientation data enhancements, we introduce a hybrid framework based on geospatial scene analysis and image processing techniques. After more accurate sensor data is obtained, we further investigate the possibility of applying sensor data analysis techniques to mobile systems and applications, such as key frame selection for 3D model reconstruction, camera motion characterization and video encoding.

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

Introduction

1.1 Background and Motivation

With today's prevalence of camera-equipped mobile devices and their convenience of worldwide sharing, the multimedia content generated from smartphones and tablets has become one of the primary contributors to the media-rich web. Figure 1.1 illustrates the most popular cameras in the Flickr Community [1]. The top 5 cameras are all smartphones. The integration of astounding quality embedded camera sensors and social capability makes the current mobile device a premier choice as a media recorder and uploader. The extreme portability also helps it to become an essential contributor to the existing large amount of user generated media contents (UGC). Moreover, nowadays an increasing number of these handheld devices are equipped with numerous sensors, *e.g.*, GPS receivers, digital compasses, accelerometers, gyros and so forth.

[1] www.flickr.com/cameras [Online; accessed Dec-2014]

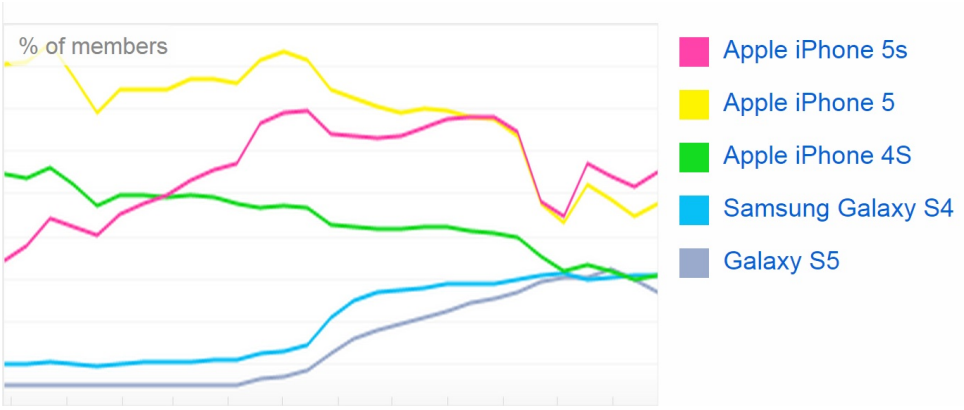


Figure 1.1: Most popular cameras trend in the Flickr community on temporal dimension (until December 2014).

Sensor information is easy to obtain by means of such trend. In addition to the media content, the success of Foursquare[2] and Waze[3] depicts the picture that these mobile devices are also actively involved in and provide massive amounts of spatial sensor data to Geographic Information System (GIS), Intelligent Transportation System (ITS) and Location-based Services (LBS) applications. Capturing, uploading and sharing of sensor information in either explicit or implicit way have become a routine part of daily life for quite a long time [112].

The usage of such sensor information has received special attention in academia as well. A growing number of social media and web applications utilize the spatial sensor information, *e.g.*, GPS locations and digital compass orientation, as a complementary feature to improve multimedia content analysis performance. Such surrounding meta-data provides contextual descriptions at a semantically interesting level. The scenes captured in images or videos can be characterized by a sequence of camera position and orientation data. Figure 1.2

[2]foursquare.com
[3]www.waze.com

Figure 1.2: Map-based visualization of a sensor-annotated video scene coverage.

illustrates the scene coverage of a video on a map, based on the associated GPS and compass sensor values. These geographically described (*i.e.*, georeferenced) media data contain significant information about the region where they were captured and can be effectively processed in various applications. A study by Divvala et al. [26] reported on the contribution of contextual information in challenging object detection tasks. Their experiments indicate that context not only reduces the overall detection errors, but more importantly, the remaining errors made by the detector are more reasonable. Many sources of context provide significant benefits for recognition only with a small subset of objects, yielding a modest overall improvement. Among the contextual items evaluated by Divvala et al., most of photogrammetric and geographic context information can be obtained from current sensors embedded in mobile devices. Slaney also studied recent achievements in multimedia, *e.g.*, music similarity computation, movie recommendation and image tagging [108]. He concludes that certain information is just not present in the signal and researchers should not overlook the rich meta-data that surrounds a multimedia object, which can help to build better feature analyzers and classifiers. Different types of sen-

sor information are also employed by various multimedia applications such as photo organization and management [29, 109, 118], image retrieval [58], video indexing and tagging [7, 104], video summarization [137, 41], video encoding complexity reduction [21], mobile video management [85, 84], street navigation systems [54], travel recommendation system [82, 35], and others.

However, the limitations of embedded sensors are also well known. For example, accuracy issues of GPS devices have been widely studied as a research topic for more than ten years. In the early stage of civilian GPS receivers, the accuracy level was very low, on the order of 100 meters or more. This was due to the fact that the U.S. government had intentionally degraded the satellite signal, a method which was called Selective Availability and was turned off in 2000. At present, the best accuracy acquired by GPS can approach 10 meters under excellent conditions. However, conditions are not always favorable due to some factors that are affecting the accuracy of GPS during position estimation such as: the GPS technique employed (*i.e.*, Autonomous, DGPS (Differential Global Positioning System) [87], WADGPS (Wide Area Differential GPS) [57], RTK (Real Time Kinematic) [56], etc.), the surrounding environmental conditions (satellite visibility and multipath reception, tree covers, high buildings, and other problems [20]), the number of satellites in view and satellite geometry (HDOP (Horizontal Dilution of Precision), GDOP (Geometric DOP), PDOP (Position DOP), etc. [113]), the distance from reference receivers (for non-autonomous GPS, *i.e.*, WADGPS, DGPS, RTK), and the ionospheric condition quality.

The accuracy issue of other location sensors, such as WiFi and cellular signal measurements (e.g., GSM), has also been extensively studied. Generally, these techniques are feasible in urban environments, but their accuracy dete-

riorates in rural areas [24]. In addition, the use of low-cost, consumer-grade sensors in current mobile devices or vehicles is another inevitable reason for the accuracy degradation.

Since some of those factors (*e.g.*, the multipath issue) cannot be eliminated with the development of GPS hardware, some post-processing algorithms and software solutions have been proposed to enhance data accuracy by a number of researchers [40, 44, 11, 1]. These methods, however, require additional sources of data to determine a more accurate position in addition to the GPS measurements, *e.g.*, Vehicular Ad-Hoc Network or WLAN information. During the GPS data collection on a smartphone, such information is not always available. Therefore, a post-processing correction method purely based on GPS measurement data itself is desirable.

Another focus of location sensor measurement correction is map matching techniques. If a mobile device collects location observations within a vehicle, the digital road network could be a key component to facilitate location data accuracy enhancement. Different from general location data, which could be measured by pedestrian-attached smartphones that travel randomly, we know for sure that the locations of vehicle-attached sensors should be observed on road arcs. Thus, map matching algorithms integrate raw location data with spatial road network information to identify the correct road arc on which a vehicle is traveling and to determine the location of a vehicle on that road arc.

In contrast to location, the accuracy of orientation data acquired from digital compasses, which is also increasingly used in many applications, has not been studied extensively. In most hand-held devices, the digital compass is actually a magnetometer instead of the fibre optic gyrocompass (as in navigation systems used by ships). Our focus is on the sensor information collected from

mobile devices along with concurrently recorded multimedia content, and hence we are interested in the accuracy of magnetometers. Generally, compass errors occur because of two reasons. The first one is *variation*, which is caused by the difference in position between the true and magnetic poles. As its name implies, it varies from place to place across the world, however, nowadays the difference is accurately tabulated for a navigator's use. In most recent mobile devices, the digital compass is able to correct this error by acquiring the current location information from the embedded GPS receiver. The second of the two errors which affect the magnetometer, *deviation*, is caused by a strong magnetic field influence of anything near the digital compass. For example, someone placing a metal knife alongside the magnetometer will cause a deflection of the compass and result in a deviation error. Steel in the construction of a building, electric circuits, motors, and so on, can all affect the compass and create a deviation error. Additionally in some regions with high concentrations of iron in the soil, compasses may provide erroneous information. Thus, when users are recording a video and collecting the direction information of a video in a building with lots of metal construction materials or in a city center with many metal cars, the digital compass devices may generate inaccurate direction values for the video content. Moreover, most of the sensors used in mobile devices like smartphones are quite low cost, which may also result in decreased accuracy. As exemplified in Figure 1.3, the red pie-shaped slice represents the raw, uncorrected orientation measurement while the green slice indicates the corrected data. As illustrated, the user is recording the tall Marina Bay Sands hotel structure towards the southeast direction, while the direct, raw sensor measurement from the mobile device indicates an east direction and hence may later lead to a completely incorrect scene expectation of a bridge (the Helix Bridge). We

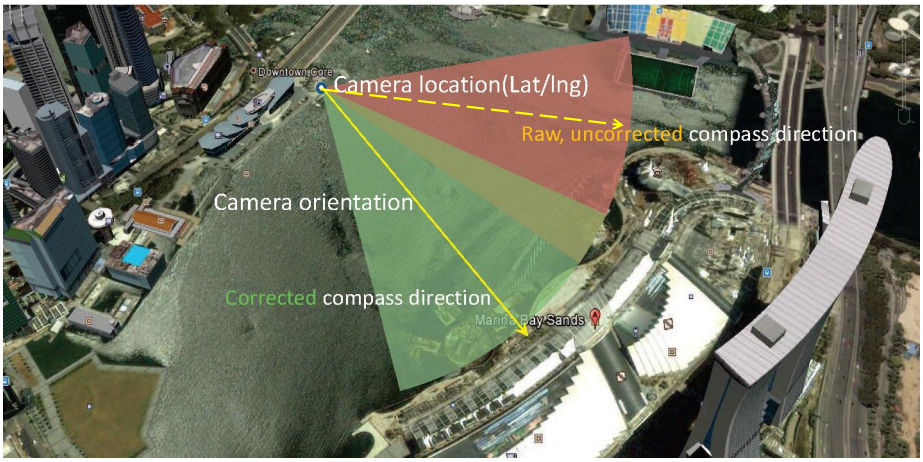


Figure 1.3: Example of a comparison of inaccurate, raw camera orientation data (red) with the ground truth (green).

found in our real world measurements that in some cases the discrepancy is more than 50 degrees from the ground-truth value. Currently, a number of existing media applications that utilize this contextual geo-information have not taken the inaccuracy problem into consideration. Thus, the algorithms that enhance the sensor data accuracy beforehand would benefit a wide range of such applications.

Given the issues outlined above, we believe that it is important and indispensable to propose effective approaches to improve the accuracy of raw sensor data collected from mobile devices.

In previously listed examples, higher level semantic results can be computed from the very low level contextual information (*i.e.*, sensor data). Here we also explore the possibility of applying sensor analysis techniques to new mobile media applications, such as video encoding improvement based on the camera motion characterization. Camera motion is a distinct feature that essentially characterizes video content in the context of content-based video analysis. It also provides a very powerful cue for structuring video data and performing

similarity-based video retrieval searches. As a consequence it has been selected as one of the motion descriptors in MPEG-7. Almost all existing work relies on content-based approaches at the frame-signal level, which results in high complexity and very time-consuming processing. Currently, capturing videos on mobile devices is still a compute-intensive and power-draining process. One of the key compute-intensive modules in a video encoder is the motion estimation (ME). In modern video coding standards such as H.264/AVC and H.265/HEVC, ME predicts the contents of a frame by matching blocks from multiple references and by exploring multiple block sizes. Not surprisingly, the computation and power cost of video encoding pose a significant challenge for video recording on mobile devices such as smartphones. Thereby, we see great potential to classify the camera motion type with the assistance from sensor data analysis and based on this intermediate result, encode mobile videos through light-weight computations.

Another application that will benefit from our sensor data analysis is the automatic 3D reconstruction from videos. Automatic reconstruction of 3D building models is attracting an increasing attention in the multimedia community. Nowadays, a large market for 3D models still exists. A number of applications and GIS databases provide and acquire 3D building models towards and from users, such as Google Earth and ArcGIS. These 3D models are increasingly necessary and beneficial for urban planning, tourism, etc. [114]. However, the adversity still lies in the fact that creating 3D objects by hand is really problematic on a large scale, especially modeling from 2D image sequences. Therefore, we leverage our spatial sensor data analysis techniques to improve the 3D reconstruction phase when the source data are videos. We explore the feasibility of using a set of UGVs to reconstruct 3D objects within an

area based on spatial sensor data analysis. Such a method introduces several challenges. Videos are recorded at 25 or 30 frames per second and successive frames are very similar. Hence not all video frames should be used — rather, a set of key frames needs to be extracted that provide optimally sparse coverage of the target object. In other words, scene recovery from video sequences requires a selection of representative video frames. Most prior work has adopted content-based techniques to automate key frame extraction. However, these methods take no frame-related geo-information into consideration and are still compute-intensive. Thus, we believe our idea with spatial data analysis is able to efficiently select the most representative video frames with respect to the intrinsic geometrical structure of their geospatial information. Afterwards, by leveraging this intermediate result — the selected key frames — the 3D model reconstruction performance can be significantly enhanced with the similar modeling accuracy.

1.2 Overview of Approach and Contributions

In this dissertation, our research focuses on how to effectively enhance the sensor data accuracy and how to utilize efficient low level sensor data analysis techniques to achieve higher level semantic results and subsequently facilitate mobile media applications. The outline of our dissertation is illustrated in Figure 1.4. We next discuss each of these issues in more details.

Usually sensor information-aided applications would directly utilize the sensor-annotated video, *i.e.*, the video content and their corresponding raw sensor data. The implicit assumption is usually that collected sensor data are correct. However, given the real-world limitations we described above, this

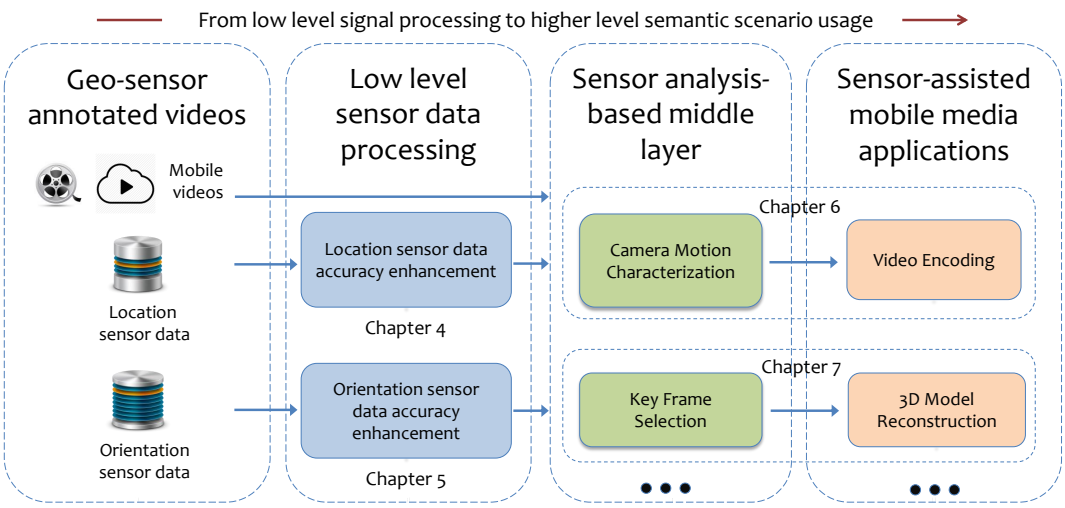


Figure 1.4: An outline of the dissertation.

assumption is generally not true. Thus, the role of our approach is to automatically and transparently process the geo data of sensor-annotated videos and then provide more accurate low level data to upstream applications. Afterwards, we analyze the processed sensor data to interpret higher level semantic information, such as camera motion types of a mobile device and representative key frames of a sensor-annotated video. Such intermediate results are later feed into mobile media applications and greatly enhance their performances.

1.2.1 Location Sensor Data Accuracy Enhancement

In sensor-annotated videos, a sequence of location measurements is recorded along with video timecode. Our approach to location sensor data accuracy enhancement contains two processing modules. For pedestrian-attached location measurements, we model the positioning measurement noise based on the accuracy estimation reported from the GPS itself, which is utilized to evaluate the uncertainty of every location measurement sample afterwards. To correct the highly unreliable location measurements, we employ less uncertain mea-

surements closely around these data in the temporal domain within the same video to estimate the most likely positions they should have. We designed two algorithms to perform accurate position estimation based on Kalman Filtering and weighted linear least squares regression, respectively. To correct vehicle-attached location measurements, we propose Eddy, a novel real-time HMM-based map matching system by using our improved online decoding algorithm. We take the accuracy-latency tradeoff into design consideration. Eddy incorporates a ski-rental model and its best-known deterministic algorithm to solve the online decoding problem. Our algorithm chooses a dynamic window to wait for enough future input samples before outputting the matching result. The dynamic window is selected automatically based on the current location sample's states probability distribution and at the same time, the matching road arc output is generated with sufficient confidence.

1.2.2 Orientation Sensor Data Accuracy Enhancement

Since the digital compasses in most current mobile devices cannot report any accuracy estimations of their direction measurements, we introduce a novel hybrid framework which corrects orientation data measured in conjunction with mobile videos based on geospatial scene analysis and image processing techniques. We report our observations and summarize several typical inaccuracy patterns that we observed in real world sensor data. Our system collects visual landmark information and matches it against GIS data sources to infer a target landmark's real geo-location. By knowing the geographic coordinates of the captured landmark and the camera, we are able to calculate corrected orientation data. While we describe our method in the context of video, images can

be considered as a specific frame of a video, and our correction approach can be applied there as well.

1.2.3 Camera Motion Characterization and Motion Estimation Improvement for Video Encoding

To address the compute-intensive challenges in camera motion characterization and video encoding, our solution is to perform sensor-assisted camera motion analysis and introduce a simplified motion estimation algorithm for H.264/AVC video encoder. From our experiments, accurate sensor data efficiently provide geographical properties which are generally quite intrinsic to device motion characterization. Moreover, in many video documents, particularly in those captured by amateurs, a global motion is commonly involved owing to camera movement and shooting direction changes. In outdoor videos, *e.g.*, videos capturing landmarks or attractions, global motion contributes significantly to the motion of objects across frames. Thus, as a key feature we only use geographic information, camera location and orientation data, to detect subshot boundaries and to infer each subshot's camera motion type from the collected sensor data without any video content processing. With generated camera motion information, we modify the HEX motion estimation algorithm used in H.264 to reduce the search window size and block comparison time for different motion categories, respectively.

1.2.4 Key Frame Selection for 3D Model Reconstruction

In the context of UGV-based 3D reconstruction, we propose a new approach for key frame selection based on the geographic properties of candidate videos.

Our technique utilizes the underlying geo-metadata to select the most representative and optimally sparse frames. Specifically, we first eliminate irrelevant frames in which the target object does not appear. The concept of geographic coverage gain is introduced and we formulate an objective function to model the geospatial difference between the original frame set and the target key frame set. A key frame subset with minimal spatial coverage gain difference is subsequently extracted by analyzing the spatial relationship among the frames based on a manifold adaptive kernel and locally linear reconstruction. In effect, our approach enables the repurposing of UGVs for 3D object reconstruction effectively and efficiently.

1.3 Organization

This thesis describes the current state of work related to the spatial sensor data processing and analysis, and the problems and issues that we have modeled and solved in this area. The remainder of this thesis is organized as follows. Chapter 2 provides a comprehensive literature survey on relevant existing work. Chapter 3 introduces the symbolic notations, and the background model to describe the viewable scene for sensor-annotated videos. Chapters 4 and 5 introduce the algorithms and systems for location and orientation sensor data accuracy enhancement, respectively. The following two mobile media applications based on spatial sensor data analysis, camera motion characterization and video encoding complexity reduction and key frame selection for 3D model reconstruction are detailed in Chapters 6 and 7, respectively. Finally, Chapter 8 concludes with a summary of the proposed research and outlines future work in this direction.

CHAPTER 2

Literature Review

This chapter presents existing research work that are relevant to our study. This review mainly focuses on four parts: location sensor data correction, orientation sensor data correction, camera motion characterization and motion estimation in video encoding, and key frame selection in 3D reconstruction.

There exist a few systems that associate videos with their corresponding geo-information. Hwang et al. [46] and Kim et al. [60] proposed a mapping between the 3D world and videos by linking objects to the video frames in which they appear. However, their work neglected to provide any details on how to use the camera location and direction to build links between video frames and world objects. Liu et al. [77] presented a sensor enhanced video annotation system (referred to as SEVA) which enables the video search for the appearance of particular objects. SEVA serves as a good example to show how a sensor rich, controlled environment can support interesting applications. However, it did

not propose a generally applicable approach to geo-spatially annotate videos for effective video search. In our prior and ongoing work [8, 6], we have extensively investigated these issues and proposed the use of videos' geographical properties (such as camera location and direction) to enable an effective search of specific videos in large video collections. This has resulted in the development of the *GeoVid* framework based on the concept of georeferenced video. The concept and framework we employed to link geospatial property to the mobile videos will be detailed in Chapter 3.

2.1 Location Sensor Data Correction

There exists some existing work to improve the location accuracy. Among many trajectory related research work and applications, map matching techniques are commonly used to employ a road network as a constraint reference for accurate location acquisition. A formal definition of map matching can be found in [12, 128] and [39]. There are several different ways to match GPS observations onto a digital map, such as geometric analysis, topological analysis, probabilistic theory and so forth. The geometry-based map matching algorithms utilize the shape of the spatial road network without considering its connectivity [12, 128]. Bernstein and Kornhauser examined three geometry matching methods: point-to-point, point-to-curve, and curve-to-curve [12]. First two methods do not make use of "historical" information and can be very unstable. In the curve-to-curve method, given a candidate node, it constructs a piece-wise linear curve from the set of paths that originates from that node. Then it calculates the distance between this curve and the curves corresponding to the network. White et al. also proposed and tested four algorithms targeting personal digital

assistant (PDA) devices [128]. The main differences consist of the utilization of heading information in their point-to-curve matching and calculating the distance between "subcurves" of equal length in their curve-to-curve matching. Since only the geometric information from the network is taken as a reference, this kind of algorithm is very efficient and scalable. However, it is unable to achieve a high accuracy and is greatly affected by measurement errors due to the same reason.

To improve the matching accuracy, some researchers proposed graph-based algorithms. They view the entire trajectory as a pure graphical curve and try to find a path (composed of a sequence of road arcs) in the road network that is as close as possible to the trajectory curve. Generally, this method employs Fréchet distance or its variants to compare these two curves [3, 17]. Alt et al. defined feasible distance measures (generalizations of Fréchet distance for curves) that reflect how close two road patterns are [3]. They abstracted the matching problem as a distance minimization problem and applied parametric search, similar as in [4] to solve it. Brakatsoulas et al. proposed two global algorithms that compare the entire trajectory to candidate paths in the road network [17]. Two similarity measures are used, Fréchet distance and weak Fréchet distance, resulting in two different map-matching algorithms which guarantee to find a matching curve with optimal distance to the trajectory. Computing the integral Fréchet distance was addressed in this work. They also addressed the performance issue and reduced the entire matching time. However, the disadvantage is also obvious. The graph-based algorithms are usually global matching procedures and have difficulty to generate arcs in real-time.

The topology-based map matching algorithms make use of the geometry

as well as the connectivity and contiguity of the road arcs in the road network [39, 97]. They leverage the topological information to reduce the candidate matches for each location sample, and develop a weighting system to measure the similarities between the geometry of a portion of the trajectory and candidate road arcs to find the most likely road arcs. Greenfeld and Joshua review several matching algorithms and propose a weighted topological algorithm [39]. They only employ the coordinates of location observations without considering the heading or speed information reported from GPS. Thus this approach tends to be very sensitive to outliers due to the inaccurately deduced vehicle headings. Especially at low speed, the uncertainty of the position information could contaminate the derivation of the heading calculated by displacement. Quddus et al. devised a weighting formula based on a priori knowledge of the statistical performance of the sensors and the topology of the network to choose the correct link [97]. They determine the vehicle position on the selected link for every two consecutive points. Their framework is simple and only uses a small number of inputs. However, this category of algorithms is very sensitive to an increase in sampling interval. The matching accuracy does degrade if two consecutive observations are not close enough to provide useful information that can be used to match the road arcs topology. A comprehensive review of 35 map matching algorithms for navigation applications since 1989 is presented by Quddus et al. [96].

Statistics-based map matching algorithms take advantage of statistical models, such as Kalman Filter [62, 94], particle filters [73], Hidden Markov Model (HMM) [13, 90, 117], etc., to solve various map matching problems. These algorithms are able to cope with noisy location measurements effectively. Kim et al. modeled the biased error of GPS into a fourth order Markov

model in order to decrease the along-track error [62]. They also reduced the cross-track error (i.e., the error across the width of the road) when the vehicle runs at a crossroad or a curved road. Their initial matching step, a point-to-curve method, is error-prone, especially in a dense urban spatial road network. Pink and Hummel incorporated vehicular motion constraints into an extended Kalman filter to improve the robustness of the matching system [94]. They also interpolated the given road network using cubic splines in the preprocessing phase and employed a Hidden Markov Model to represent road network topological constraints. In addition to the road network topology, Billen et al. included several features into the matching process using HMM, such as position history and orientation history, which considerably increase the classification robustness [13]. Liao et al. used a hierarchical Markov model to learn and infer a user's daily movements in an urban environment. They proposed to bridge the gap between location sensor data and high-level semantic information base on a multi-level abstraction. At the signal level, they employed Rao-Blackwellized particle filters (RBPF) for posterior estimation. Newson and Krumm also proposed a HMM-based map matching framework, where the main difference from others are the intuitive transition probability setting based on the discovered pattern from their collected trajectory data [90]. They also make their GPS data, ground truth, and relevant road network publicly available to facilitate the fair comparison of other map matching algorithms. Similar to Newson's work, Thiagarajan et al. also performed a quantitative evaluation of the end-to-end quality of time estimates from noisy and sparsely sampled locations [117]. They collected a wardriving database for low accuracy WiFi localization data, and discuss the accuracy-energy tradeoff between using WiFi location data and GPS samples. However, only a few studies have focused on

the real-time decoding issue of the HMM model. Goh et al. proposed a variable sliding window scheme to provide an online solution while the delay bound of the road arc generation is not guaranteed [37]. Additionally, the tradeoff relation between the accuracy and latency from online decoding strategies has not been extensively studied yet.

In case of pedestrian-attached location sensor correction, the limitation of these map matching techniques is that the positioned object has to move along the road map, since the digital road network is considered as the only feasible path. Thus, existing approaches mostly target vehicle positioning tasks or vehicle navigation systems, while our approach processes location data generated from any free movement. The raw location data as the input of our pedestrian-customized system can be generated from the movement of cars, bikes, people, etc. Our algorithms are able to improve the accuracy of those trajectories that are not necessary to have corresponding roads.

In addition to map matching techniques, researchers also leverage multiple information fused together to obtain more accurate locations. Hii and Zaslavsky combine WLAN positioning and acoustic localization techniques to improve the location accuracy [44]. Bell et al. validated Wireless Access Points (WiFi APs) for determining location in their study [11]. Otsason et al. presented a GSM indoor localization system for large multi-floor buildings [92]. However, these information sources also have inevitable noises (the accuracy of WiFi and GSM localization technologies are around 40 meters and 400 meters respectively [24]). In data fusion approaches, to form hierarchical and overlapping levels of sensing, the Kalman filtering method has been widely applied to GPS navigation processing [53, 99]. However, those approaches all need additional data sources coupled with GPS locations. Most of them acquire information from Inertial

Navigation Systems (INS) for autonomous mobile vehicles, which consist of motion sensors (accelerometers) and rotation sensors (gyroscopes). As a result, their applications are also limited to vehicle location-aware systems, e.g., Intelligent Transportation System (ITS). In our case, we adopt the Kalman filtering method to improve the location accuracy without any assistance from other sources of information, but purely based on the measured data acquired from GPS receivers in smartphones. The moving "object", not limited to vehicles, could be anyone who holds the positioning sensor.

2.2 Orientation Sensor Data Correction

Researchers have leveraged various content-based computer vision techniques to estimate the viewing direction of photos. They geo-locate a photo and estimate the camera orientation by registering the image onto street level panoramas [64], Google Street View and Google Earth Map [93]. In the image matching process, feature matching happens for every candidate image individually, which imposes a high computational cost and makes real-time applications unfeasible. Luo et al. utilized a Scale-Invariant Feature Transform (SIFT) flow to match a photo in a database followed by image geometry calculation, to determine and filter the viewing direction [83]. However, these methods can be applied only to individual photos and cannot be easily applied to video applications. Moreover, they all require either a constrained camera location (since a street view is only applicable for photos taken on or near a road network) or a relatively large image database (even satellite images) to perform the matching phase.

In addition to the absolute viewing direction estimation, other research work also look into the relative camera orientation calculation problem, which

is part of extrinsic camera calibration (deciding the positions and orientations of the camera) [9, 49, 127]. However, these methods can only report the *relative* angle between the main object in the image and the camera, while our target is to estimate the real orientation (values with semantic meanings, such as north, east, south and west).

Recently, the Structure from Motion (SfM) technique has been extensively exploited to reconstruct 3D models from a collection of images [48, 70, 71, 100]. SfM estimates three-dimensional structures from two-dimensional image sequences which may be coupled with local motion signals. A set of images that show an object from different directions are registered to 3D scenes by feature point matching and the camera pose (including location and orientation) of each image is estimated by image geometry calculation. Thus the camera viewing orientation can be extracted from the camera pose parameters as one output of the SfM procedure. However, the scene models are usually reconstructed from the datasets at a scale of 10^3 to 10^5 photos acquired via text-based search from the web or purposely captured [76]. Since these algorithms were not devised for a dedicated sensor data correction purpose, they ignore all contextual geo-information. As a result, the preliminary dataset requirements and extensive processing time make these methods unsuitable for large-scale or real-time camera orientation correction.

2.3 Camera Motion Characterization and Motion Estimation in Video Encoding

Several approaches have been developed to estimate camera motion based on the analysis of the optical flow computed between consecutive images [51, 25, 15]. Jinzenji et al. employed Hermart transform coefficients to describe camera motions, including scaling, rotation and translation [51]. They proposed a new scheme to produce layered sprites throughout a video shot with separated background and foreground information. Denzler et al. applied statistical methods which are based on the normal optical flow field [25]. In order to avoid an inefficient global search, they divided the scenes into regions and extracted features from a sparse normal optical flow field to train a Gaussian-distribution classifier and a Kohonen feature map. Their model classified unknown camera motion into nine classes based on different pan-tilt movements. Bouthemy et al. estimated a 2D affine motion model between pairs of successive frames accounting for the globally dominant image motion [15]. It detects both cuts and progressive transitions. The significance of each component of the estimated global affine motion model provides a qualitative description of the dominant motion. However, the estimation of the optical flow, which is usually based on gradient or block matching methods, is computationally expensive [50]. Moreover, when the camera moves fast, there will be significant displacement between consecutive frames, which may lead to an inaccurate estimation of the optical flow.

Considering that most videos are not provided in the form of image sequences, but rather as compressed formats, some approaches directly manipulate MPEG-compressed video to extract camera motion using the motion vec-

tors as an alternative to the optical flow [126, 5, 59, 30, 43]. Wang and Huang proposed a variation of the least-square principle that rejects outliers at each iteration by using a Gaussian distribution to model how well the global motion parameters match with the motion field [126]. Ewerth et al. also presented an outlier removal algorithm by checking the change smoothness and the number of supporting motion vectors from the neighborhood blocks [30]. They clearly distinguished the translational and rotational camera motions. Their system also directly worked on motion data available from the compressed video stream. Ardizzone et al. clustered the motion vectors in the compressed domain and individuated the dominant regions for segment feature extraction [5]. Kim et al. fit the motion vectors from an MPEG stream into a 2D affine model to detect camera motions [59]. They filtered out noises and normalized various types of motion vectors. Camera motions and segment boundaries are obtained by interpreting the estimated model parameters and the homogeneity within each unit. Heuer and Kaup proposed to perform linearization of the sine and cosine terms in the affine model to make the parameter estimation both efficient and reliable [43]. Nonparametric motion models have also been proposed in the motion feature space [28]. Nevertheless, the MPEG motion vectors estimated by video encoders are not always consistent with the actual movement of macro-blocks since many of them correspond to the movements of foreground objects. Thus, the effectiveness of these methods relies on their preprocessing stages to reduce the influence of irrelevant motion vectors. When the video contains significant camera or object motions, such irrelevant motion vectors may be prevailing and interfering with the preprocessing stages. Furthermore, accurately detecting camera zoom operations is difficult because of the noise in motion vectors due to independent object motions in a frame or MPEG encod-

ing properties, such as quantization errors, and other artifacts. Hence, these methods usually only work well for videos with special encoding formats.

Lertrusdachakul et al. [68] analyzed camera motion by processing the trajectories of Harris interest points that are tracked over an extended time. However, when the camera moves fast and the background content changes rapidly, interest points in the background may not be tracked for long. Additionally, the Harris interest point detector is not invariant to scale and affine transforms, which may be significant between consecutive frames when the camera moves fast. Battiato et al. [10] used motion vectors of SIFT features to estimate the camera motion in a video, but inaccurate results were prone to be generated with their approach since foreground and background features are treated without discrimination.

In video encoding, the key to the significant temporal compression is motion estimation, which seeks to identify blocks in a frame that match those in a reference frame at different – but close – locations. To exploit the sensor information for an efficient video encoding purpose, Hong et al. [45] proposed an accelerometer-assisted model to simplify the motion estimation part in the encoder. However, the authors only considered the horizontal and vertical movements of the camera. Their experimental evaluations are based on MPEG-2, which is no longer a state-of-the-art compression technique. Another sensor-assisted motion estimation algorithm proposed by Chen et al. [22] employed additional digital compass information and measurements were obtained with H.264/AVC. Nevertheless, their work is still limited to rotational camera movements. Both the above algorithms cannot handle linear camera movements, which is very common in video clips taken by handheld devices. Furthermore, the sensor information utilized by those algorithms only leveraged accelerom-

eter and compass information, while there are other sensors available, which could also improve the efficiency of video encoding. In addition, the European patent application EP1921867 presents an idea of using vehicle movement information to assist in video compression [121]. However, this method focuses on vehicle motion and a vehicle-mounted camera, and provides no implementation or evaluation.

2.4 Key Frame Selection for 3D Model Reconstruction

3D model reconstruction from images [36, 69, 33, 72] or videos [75, 91, 23] has been of wide interest to the research community. Shum et al. [106] exploit the information redundancy in images by using two virtual key frames to represent a sequence, which indicats the importance of a key frame extraction procedure. Other researchers [2, 89, 103, 102] have considered selecting key frames from a video prior to initiating the reconstruction process. The existing selection techniques extract key frames from one video source, while we propose selection techniques from multiple crowdsourced UGVs. In the method from Ahmed et al. [2], the selection mechanism of key frames is based on a) the number of frame-to-frame point correspondences obtained from a geometrically robust information criterion (GRIC) [120], and b) the point-to-epipolar line cost for the frame-to-frame correspondence set. Other work [102] considers more factors to select key frames: the ratio of the number of point correspondences found to the total number of point features found, the homography error, and the spatial distribution of corresponding points over the frames. Seo et al. [103] use the

ratio of the number of correspondences to the total number of features found. When given an image sequence, Pollefeys et al. [95] select key frames based on a motion model selection mechanism explored by [119]. They select key frames only if the epipolar geometry model explains the relationship between the pair of images better than the simpler homography model and all degenerate cases are discarded.

Similarly, in real time localization and 3D reconstruction or visual Simultaneous localization and mapping (SLAM) systems, Mouragnon et al. [89] take a new key frame if the number of matched points with the last key frame is not sufficient or the uncertainty of the calculated camera position is too high. Zhang et al. [135] employ five representative techniques in the content-based image retrieval (CBIR) field for key frame detection to compare several performance metrics in their systems. In Klein and Murray's work, key frames are added whenever the following conditions are met: a) the tracking quality is good; b) the time since the last key frame was added exceeds twenty frames; and c) the camera is a minimum distance away from the nearest key point on the map [63]. Dong et al. [27] extract key frames from all reference images to abstract the space with a few criteria: a) the key frames should be able to approximate the original reference images and contain as many salient features as possible; b) the common features among these frames are minimal in order to reduce the feature non-distinctiveness in matching; and c) the features should be distributed evenly in the key frames such that given any new input frame in the same environment, the system can always find sufficient feature correspondences and compute accurate camera parameters. One of the common characteristics of the existing techniques is that they select key frames depending on different geometric models to score the correspondence of match-

ing points between frames. All these methods focus on the frame content- or point cloud-level processing which are still compute-intensive. Our method instead focuses on UGV attached sensor data to choose the most representative key frames in geographic space.

Mordohai et al. [88] also used GPS data within a real time 3D reconstruction approach from videos that makes use of location information to place the reconstructed models in geo-registered coordinates on maps. However, their acquisition system needs to be fully customized and they simply select the candidate frames whose baseline between two consecutive frames exceeds a certain threshold for further 3D reconstruction. They also mention that the threshold varies depending on the different objects' scene depth. Instead, we employ GPS information and more sophisticated algorithms to select a set of geographically representative frames of the collected videos. To the best of our knowledge, there exists no prior method that leverages crowdsourced videos that are contextually enriched at a very fine-grained level and extracts key frames based on their geographic characteristics to reconstruct 3D models.

CHAPTER 3

Preliminaries

This chapter introduces a basic model that describes the viewable scene in videos. The concept and framework we employed to link geospatial property to the mobile videos are presented here.

The geo-sensor data utilized in our approach consists of a series of contextual descriptions of mobile video content that reflects the geospatial properties of the scenes it captures (as illustrated in Figure 1.2). To allow users to conveniently acquire geo-tagged videos, we leverage two custom recording apps, *GeoVid* [101] and *MediaQ* [61], publicly available for both Android and iOS. When a user begins to capture a video, the GPS and compass sensors start to *continuously* record location and orientation information of the (moving) camera. All the collected sensor data (*i.e.*, camera location and orientation, the corresponding frame timecode and video ID) are combined into a JSON format and uploaded to a portal, to which users can also submit various spatial and

contextual queries, browse, and retrieve the videos with their sensor data via web APIs[1].

We adopt the field-of-view (FOV, also called the *viewable scene*) model introduced by Arslan Ay et al. [8]. An FOV describes a scene area captured by a camera positioned at a given location. The description of a camera's viewable scene consists of three parameters: the camera location $\mathcal{L}$, the camera orientation θ, and the viewable angle α (defined in Equation 3.1). The camera position $\mathcal{L}$ is composed of latitude and longitude coordinates provided by a positioning device (*e.g.*, GPS receiver) and the camera orientation θ is obtained based on the direction angle value from a digital compass. The viewable angle α is calculated based on the camera and lens properties at the current zoom level.

$$FOV \equiv \langle \mathcal{L}, \theta, \alpha \rangle \tag{3.1}$$

With this model and the application, we are capable of collecting and managing sensor data in conjunction with video contents during the recoding phase. Note, each mobile device model may use different sampling frequencies for different sensors. Ideally we acquire one *FOV triplet* per frame. If that is not feasible and the granularity is coarser due to the device limits, we perform linear interpolation to generate triplets for each frame.

In Table 3.1, we also briefly present the important symbols and their meanings used in this thesis.

[1] http://api.geovid.org

Symbol	Unit	Meaning
l	latitude, longitude	original reading from localization sensors
a		accuracy level value of each GPS reading
θ	degree	original reading from orientation sensors
τ		processed data
g		ground truth data
L/L'		a sequence of original/ processed location data
S/S'		a sub-sequence of original/ processed location data
E/E'		average error of original/ processed data
e		road arc
m	latitude, longitude	match point
t/t'	s	location measurement timecode/ map matching output timecode
$\mathcal{M}$		emission probability
$\mathcal{T}$		transition probability
π		initial probability
δ		hidden state probability
ψ		backtracking pointer of the selected hidden state
$\mathcal{H}$		accuracy penalty entropy
γ		the parameter to control the tradeoff between accuracy and latency
$\mathcal{O}$		visible buildings
$\mathcal{P}$		set of all frame geo-location data points
$\mathcal{K}$		key location set
$\mathcal{K}$		kernel function
G_{diff}		the average expected square coverage gain difference

Table 3.1: Summary of symbolic notations.

CHAPTER 4

Location Sensor Data Accuracy Enhancement

4.1 Introduction

In this Chapter we present two frameworks for pedestrians-attached and vehicle-attached location sensor data, respectively. Since the pedestrian-attached location sensor may not travel along the road network, we model the positioning measurement noise purely based on the accuracy estimation reported from the GPS itself, which is utilized to evaluate the uncertainty of every location measurement sample afterwards. To correct the highly unreliable location measurements, we employ less uncertain measurements closely around these data in the temporal domain within the same video to estimate the most likely positions they should have. On the other hand, for vehicle-attached location data, we

employ and improve the map matching techniques. Our framework associates a sorted list of position data to the road network on a digital map in realtime. Both the accuracy and latency of the map matching results outperform the existing methods.

4.2 Location Data Correction from Pedestrian Attached Sensors

First, to have a better understanding of the noisy sensor data and their typical error patterns, we collected and carefully examined more than 80 mobile videos associated with sensor information, which are publicly available from the Geovid website [1]. We report our observations and summarize some typical inaccuracy pattern that emerged in those real sensor data.

4.2.1 Observation of Real Sensors

To evaluate those location coordinates sequences, we display every location measurement on a map interface and compare its coordinates with the ground truth position. From manually analyzing those inaccurate location measurements' properties in GPS data sequences, we summarize two typical error patterns of location measurements.

- *Extreme inaccuracy at start.* A standalone GPS system needs orbital information of the satellites to calculate the current position, which provides the first position in approximately 30-40 seconds. To avoid empty measurements in location data collection, many mobile devices combine A-

[1] http://api.geovid.org/

GPS [65], other location services including Wi-Fi Positioning System and cell-site triangulation, and sometimes a hybrid positioning system [134] to improve the startup performance of GPS receivers. However, those assistant systems require additional network resources to provide locations and still use the satellites in poor signal conditions. From our observations, since the network resources are not always available when users start recording the video, some location data generated by GPS system at the beginning of a location sequence file are extremely noisy.

- *Sudden moderate inaccuracy.* GPS operation uses radio signals from satellites. In very poor signal conditions, for example in a city, these signals may suffer multipath propagation where signals bounce off buildings, or are weakened by passing through atmospheric conditions, walls or tree cover. Thus, when users encounter these conditions during video recoding, some GPS navigation devices without network connections may not be able to work out a position due to the fragmentary signal, rendering them unable to function until a clear signal can be received again. As a result, we observe some sudden moderate incorrect location measurements generated in the middle of some location sequence files.

4.2.2 Problem Formulation

To filter out such noises, we employ two post-processing methods. Here we begin by describing the problem in a formal way.

Problem Statement: Given a sequence of positions L and their corresponding timestamps and accuracy measurement sequence T and A, find a sequence of estimated position coordinates, $F : l_i \rightarrow \tau_i$, such that the processed

position sequence $L' = \{\tau_1, \tau_2, ..., \tau_n\}$ is more accurate in the sense of comparing E_L with E'_L, where $E'_L = \frac{1}{n}\sum_{i=1}^{n} \delta'_i$ and δ'_i is each processed position's distance to the ground truth. Each τ_i also includes an updated longitude x'_i and latitude y'_i.

An original GPS reading l_i always comes with an accuracy measurement value a_i. The accuracy sample indicates the degree of closeness between a GPS measurement l_i and its true, but unknown position g_i. If a_i is relatively high, it means that the actual position g_i is far away from l_i. We utilize the model of location measurement noise with l_i and a_i [74], where the probability of the real position data is assumed to be normally distributed with a mean of l_i and its standard deviation σ_i. We then set $\sigma_i^2 = g(a_i)$, where the function g is monotonically increasing.

Let a small sub-sequence of a given GPS dataset (termed **GPS segment** or segment for short) be $S_\kappa = \{l_i, l_{i+1}, \cdots, l_j\}$. It has a relatively short duration and its moving speed v_κ is assumed constant. The original GPS readings can also be expressed as a series of disjoint segments $L = \{S_1, S_2, ..., S_m\}$ with their corresponding velocity $V = \{v_1, v_2, ..., v_m\}$. For a given segment S_κ, we can estimate the accurate position τ_k based on the fusion of two sources of data: (1) the measurement l_k with noises directly from the GPS receiver and (2) the displacement calculation based on the last estimated position τ_{k-1}, the velocity v_κ in this segment, and the time duration between t_k and t_{k-1}.

Both, the noisy measurement data and approximations in the uniform motion model of each segment, however, introduce some uncertainty about the inferred value of a position. To better estimate the position closer to the real coordinates, it will be reasonable to trust values with a smaller estimated

uncertainty more than those with a larger one.

We propose two post-processing methods: a Kalman Filtering-based method (detailed in Section 4.2.3) and a weighted linear regression-based method (detailed in Section 4.2.4). These two methods have different assumptions on the corrected trajectory. They assume a constant velocity and a linear movement on each segment, respectively.

4.2.3 Kalman Filtering based Correction

We model the process in accordance with the framework of the Kalman filter. It operates recursively on two streams of noisy data to produce an optimal estimate of the underlying positions. We describe the position and velocity of the GPS receiver by the linear state space:

$$\pi_k = \begin{bmatrix} x_k & y_k & v_{\kappa x} & v_{\kappa y} \end{bmatrix}^T,$$

where $v_{\kappa x}$ and $v_{\kappa y}$ are the longitude and latitude component of velocity v_κ. In each segment S_κ, v_κ can be estimated by some less uncertain coordinates and their timestamp information. We define the state transition model F_k as

$$F_k = \begin{bmatrix} 1 & 0 & \Delta t_k & 0 \\ 0 & 1 & 0 & \Delta t_k \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix},$$

where Δt_k is the time duration between t_k and t_{k-1}. We also express the observation model H_k as

$$H_k = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}.$$

H_k maps the true state space into the measured space. For the measurement noise model, we use a_k to present the covariance matrix R_k of observation noise as follows:

$$R_k = \begin{bmatrix} g(a_k) & 0 \\ 0 & g(a_k) \end{bmatrix}$$

Similarly, Q_k can also be determined by a diagonal matrix but using the average of $g(a_\delta)$, whose corresponding position coordinates l_δ and timestamp t_δ were used to estimate v_κ in this segment.

We apply this process model to the recursive estimator by two alternating phases. The first phase is the "prediction", which advances the state until the next scheduled measurement is coming. Second, we incorporate the measurement value to update the state.

As reported by our real data observations, in many cases, the accuracy measurements at the start of a GPS data sequence are worse than those at the end. To efficiently correct those spotty GPS readings with significant uncertainty, we start processing position data in a reverse way (*i.e.*, from l_j to l_i in segment S_κ) with our recursive algorithm. Finally, after processing each GPS segment S_κ from L, we obtain a series of updated position sequence segments $S'_\kappa = \{\tau_i, \tau_{i+1}, ..., \tau_j\}$. The corrected result is composed of this series of segments, $L' = \{S'_1, S'_2, ..., S'_m\}$.

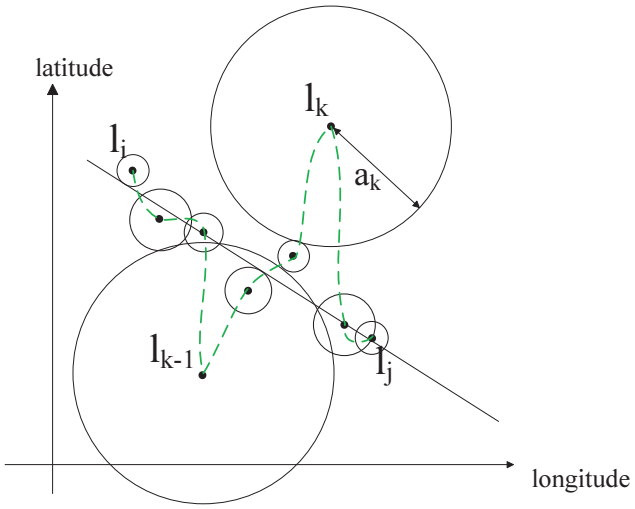


Figure 4.1: Visualization of weighted linear least squares regression based correction model. GPS samples in 2D dimension (green line indicates the original trajectory).

4.2.4 Weighted Linear Least Squares Regression based Correction

The second correction model is based on a piecewise linear regression analysis. Since we post-process the GPS sequence data, we can fully utilize both previous and future GPS readings, from l_i to l_j, to estimate the current position τ_k, where $i < k < j$. The piecewise linear regression model computes several estimated position data sequences S'_κ, which can later be linked into an integrated sequence as corrected output L'.

Within a segment S_κ, different GPS readings contain varying accuracy measurements. Figure 4.1 illustrates the concept of the linear regression for one segment S_κ. For example, position l_i has $a_i = 5$ meters, and another position l_k has $a_k = 600$ meters. This indicates that these regressors have been observed with certain errors and those errors have varying variances. Thus, when calculating regression estimator for S_κ, the contributions of different points with

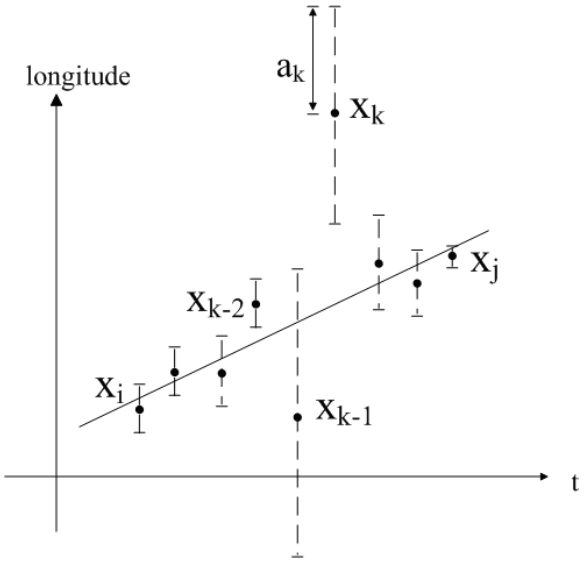


Figure 4.2: Visualization of weighted linear least squares regression based correction model. GPS samples in the longitude dimension.

varying accuracy measurements to the regression line should be unequal. With the assumption that these errors are uncorrelated with each other and with the independent variables l_i, we utilize the weighted least squares method to generate estimators $\hat{\beta}_\kappa$ for each S_κ, instead of standard regression models.

Since x_i and y_i are two independent variables, we estimate model function parameters $\hat{\beta}_\kappa$ for longitude and latitude values with respect to time separately. We illustrate how the weighted least squares method works for the longitude values regression in Figure 4.2. The goal is to find $\hat{\beta}_\kappa$ for the model which "best" fits the weighted data. By using the weighted least squares method, we need to minimize R, where

$$R = \sum_{k=i}^{j} W_{kk} r_k^2, \quad r_k = x_k - f(t_k, \hat{\beta}_\kappa) \tag{4.1}$$

Here r_k is the residual defined as the difference between the original measured longitude value and the value predicted by the model. The weight W_{kk}

is defined as:

$$W_{kk} = \frac{1}{\sigma_k^2} \tag{4.2}$$

Here σ_k is the deviation of the measurement noise. Because it is proved that $\hat{\beta}_\kappa$ is a best linear unbiased estimator if each weight is equal to the reciprocal of the variance of the measurement. As described in Section 4.2.2, we modeled the measurement noise as a normal distribution with mean x_k and standard deviation $\sigma_k = g(a_k)$ in the longitude dimension.

Base on this model, measurements x_k with a high a_k value, which indicates high uncertainty, will not have much impact on the regression estimation. Usually, these uncertain measurements reflect many jumping GPS locations, which are far away from where the real positions should be. Considering the regression line is estimated mostly by the confidence data and these data are almost consecutive in the temporal domain, we are able to correct those spotty GPS locations to positions that are much closer to the real coordinates. Thus, after we calculate the regression line $\hat{\beta}_\kappa$, we update the longitude values based on the following rules:

$$\begin{cases} x'_k = x_k, & \text{if } a_k < TH_a \text{ and } a_k < r_k \\ x'_k = f(t_k, \hat{\beta}_\kappa), & \text{else} \end{cases} \tag{4.3}$$

Here TH_a is the accuracy measurement threshold. We only update the latitude value if its corresponding accuracy is measured as being considerably uncertain, or its distance to the projection point on the regression line is less than its accuracy measurement value (e.g., the latitude value x_i and x_k in Figure 4.2 will be updated in S'_κ, while $x'_{k-2} = x_{k-2}$). We do not use projection points on the

regression line for all x_k, because some original longitude values have relatively high confidence with their measurements, and keeping those extremely confident GPS raw data while updating the remaining uncertain data will generate a more accurate position sequence according to the observation that the GPS carrier may not move following a standard linear model.

Finally, by computing every longitude value x'_k in each piece S'_κ, we can link every approximate linear piece together and obtain an updated position sequence L'.

4.3 Location Data Correction from Vehicle Attached Sensors

For the location data reported from vehicle attached sensors, we utilize the road network information and map matching techniques to enhance the positioning accuracy. There exist a number of statistical matching approaches that unfortunately either process trajectory data offline or provide an online solution without an infimum analysis. Here we propose a novel statistics-based online map matching algorithm called Eddy with a solid error- and latency-bound analysis. More specifically, Eddy employs a Hidden Markov Model (HMM) to represent the spatio-temporal data as state chains, which elucidates the road network's topology, observation noises and their underlying relations. After modeling, we shape the decoding phase as a ski-rental problem, and propose an improved online-version Viterbi decoding algorithm to find the most likely sequence of hidden states (road routes) in real-time. We reduce the candidate routes search range during the decoding for efficiency reasons. Moreover,

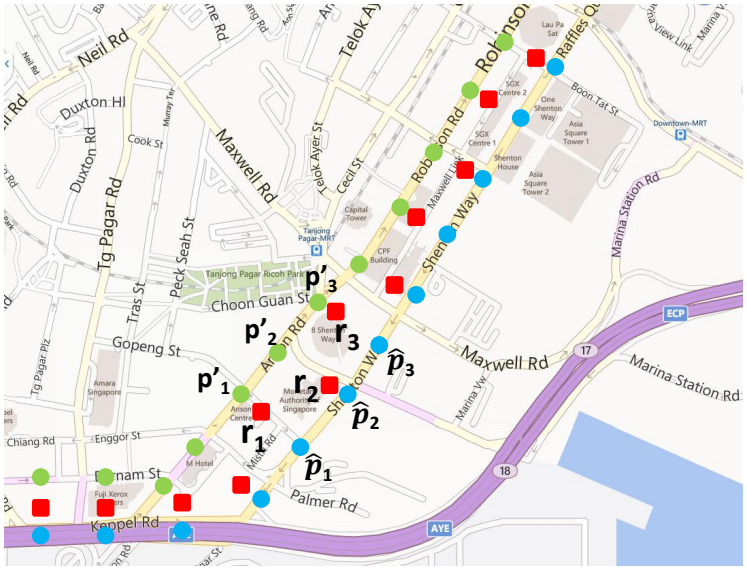


Figure 4.3: Illustration of the map matching problem.

our deterministic decoder trades off latency for expected accuracy dynamically, without having to choose a fixed window size beforehand. We also provide the competitive analysis and proof that our online algorithm is error-bounded (with a competitive ratio of 2) and latency-bounded.

The input of a typical map matching algorithm is a temporal sequence of location points, *i.e.*, a trajectory. In practice, most raw location information provided from sensors is not highly accurate or not easily interpretable. Therefore, a map matching algorithm is desirable to help improve the positioning accuracy if the respective digital map is reliable, and to associate the coordinates with the surrounding spatial entities seamlessly. The map matching problem is illustrated in Figure 4.3. The red dots such as r_1, r_2 and r_3 are the measured raw location coordinates. The task of map matching is to find the true roads that the moving object is on. As illustrated, it could be a challenging problem since either the green-dots trajectory (p'_1, p'_2 and p'_3) or the blue-dots trajectory ($\hat{p}_1, \hat{p}_2$ and $\hat{p}_3$) can be the actual driving path, and it is impossible

to tell by only analyzing separate samples. A number of statistics-based map matching algorithms were proposed and developed in recent years. They employed statistical models to solve various map matching problems and distinctly showed the ability to cope with noisy GPS measurements effectively. Particularly, algorithms based on a Hidden Markov Model (HMM) and its variants have been adopted due to their capabilities of concurrently evaluating multiple hypotheses during the mapping procedure [79, 133, 90, 31]. They have also been proven to be tolerant against highly noisy observations, *e.g.*, the location fingerprints from GSM towers [116], and the accuracy degradation owing to the increase of a trajectory's temporal sparseness [90, 79].

In the context of Markov information sources and hidden Markov models, the Viterbi algorithm, a dynamic programming algorithm, is widely used for decoding such models. This algorithm finds the most likely sequence of hidden states for the given observation sequence [122]. It computes a forward pass over the input sequence to compute probabilities, followed by a reverse pass to compute the optimal state sequence. Therefore, all the data must be obtained before any of the hidden states can be inferred. The result of the underlying state chain is called a *Viterbi path*. However, when applied to a real-time or an interactive system, one noticeable disadvantage of the Viterbi algorithm is that the optimal state sequence cannot be computed until the entire input has been observed.

For latency-sensitive applications such as route navigation and traffic incident detection, it is unacceptable to receive map matching results, *e.g.*, on which road arc the truck is driving, after the whole itinerary is finished. In HMM-based map matching, the key input and output of a traditional Viterbi decoder are the location observations (*e.g.*, GPS measurements) and the most

likely road trajectory of a moving object. Conceptually, the input observation stream could be extremely long, or even infinite, which leads to a significantly longer latency than a timely response that systems may require. Therefore, the traditional Viterbi decoder is not suited for real-time applications where there are strong latency constraints.

Meanwhile, accuracy is also another crucial factor for most location-based applications. To shorten the mapping delay, a system has the freedom to match raw location measurements greedily, mapping each sample immediately as an extreme case, without waiting for enough future observations. However, it is undesirable to give up the accuracy increase gained by map matching techniques or even worse, pick an incorrect road path as output. The risk of selecting a false road may cause serious issues in real system such as incident detection. Any inaccurate output also raises the expected monetary cost in some enterprise services, *e.g.*, logistics truck monitoring, fleet scheduling and others. Thus, an intelligent algorithm which understands and wisely practises the balance between accuracy and latency is desirable.

Here we propose Eddy, a novel real-time HMM-based map matching system by using our advanced online decoding algorithm. We take the accuracy-latency tradeoff into design consideration. Our algorithm chooses a dynamic window to wait for enough future input samples before outputting the matching result. The dynamic window is selected automatically based on the current location sample's states probability distribution and at the same time, the matching output is generated with sufficient confidence. Our contributions in this work include:

- An improved real-time HMM-based map matching system is presented

which is novel with respect to its tradeoff analysis and dynamic window selection algorithm during the decoding phase.

- A competitive analysis and proof illustrating that our online decoding algorithm is error-bounded (with competitive ratio of 2) and latency-bounded.

- Accuracy evaluation and latency comparison between our map matching system results and existing online decoding algorithm outputs.

4.3.1 HMM-based map matching

We first give the preliminaries and the formal definitions of the map matching problem using HMM.

Definition 1 *(Road Network)* : A road network $G(V, E)$ represents a finite street system which consists of a set of one-way or two-way road curves, called *road arcs*, in 2D Euclidean space. Each road arc e_i ($e_i \in E$) is assumed to be piecewise linear and can be characterized by a finite sequence of points $A^i = (a_1^i, a_2^i, ..., a_m^i)$. The end points here a_1^i and a_m^i are nodes and belong to the vertex set V. Other points in the middle are referred to as shape points and each e_i has some properties such as speed constraints.

Definition 2 *(Location Trajectory)* : A location trajectory $L = \{l_1, l_2, \ldots, l_n\}$ is a sequence of measurements from localization sensors (such as GPS) according to the time sequence $T = \{t_1, t_2, \ldots, t_n\}$. Each position measurement l_i consists of a coordinate, *i.e.*, longitude x_i and latitude y_i. We further denote the ground truth of the position sequence data as $G_l = \{g_1, g_2, \ldots, g_n\}$ and their belonging road arcs $G_e = \{\gamma_1, \gamma_2, \ldots, \gamma_n\}$, $G_e \in E$.

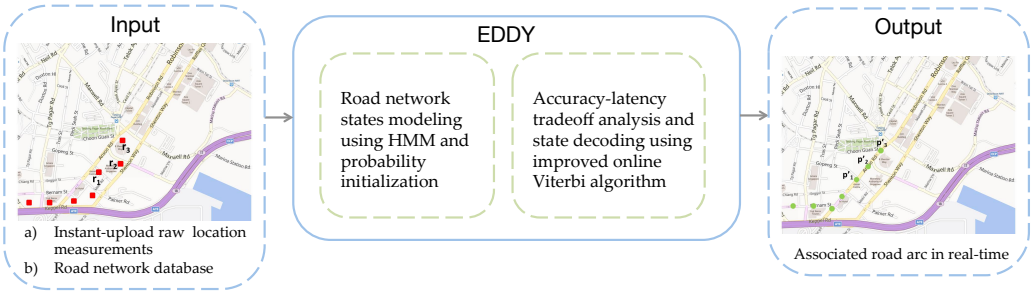


Figure 4.4: System overview of Eddy.

Definition 3 *(Match Point)* : The match point m_i^j of a location measurement sample point l_i on a road arc e_j is the point that $m_i^j = argmin_{\forall m_k^j \in A^j} dist(m_k^j, l_i)$, where $dist(m_k^j, l_i)$ returns the great circle distance between l_i and any point on A^j, including end points and shape points.

Problem Statement: Given the road network $G(V, E)$, and the trajectory information L and T, find the most likely path $P = \{p_1, p_2, \ldots, p_n\}$, where $a_m^{i-1} = a_1^i$ and $P \subset E$, which is a subset of connected road arcs from G, along with each p_i's mapping output time $T' = \{t'_1, t'_2, \ldots, t'_n\}$.

Figure 4.4 illustrates the system overview of Eddy. It takes the location measurements and road network databases as input. The positioning data should be instantly uploaded since we focus on the latency-sensitive applications and services in this study. Eddy results in a real-time streaming of road arcs with guaranteed accuracy and latency level.

Our system consists of two parts. We first present the road arc traveling problem as a hidden states transition model. Based on the framework of HMM, the random variable e_t and l_t are a hidden state and an observation at time t, respectively (see the state transition flow in Figure 4.5). In the context of the map matching problem, we model every road arc e_i as a hidden state and each location measurement l_t as an observation emitted by the hidden state.

Two types of arrows in this figure (horizontal and vertical arrows) indicate two important parameters in the model. The horizontal arrow represents the *transition probability* between two consecutive hidden states. It quantifies the likeliness that a vehicle is moving from road e_{t-1} to road e_t. Each vertical arrow represents the *emission probability* between the hidden state and the observation. It represents how likely the measurement l_t can be observed if the vehicle is driving on a certain road arc.

The second part is the online Viterbi decoding algorithm (see the trellis in Figure 4.5) which is improved based on our quantitative accuracy-latency tradeoff analysis. During the decoding phase, candidate arc paths are sequentially generated and evaluated on the basis of their likelihoods. Our goal is to find the maximum likelihood path over the Markov chain that has the highest joint emission/transmission probabilities and still holds the latency bound.

Formally, the map matching problem is modeled by transition, emission and initial probability :

$$\lambda = (\mathcal{T}, \mathcal{M}, \pi)$$

The state set is E and the observation set is L. In our model, the initial probability π_i of being in state e_i is defined as the emission probability at this state. The emission probability $\mathcal{M}_i(l_t)$ of observation l_t from state e_i is obtained by modeling the positioning measurement noise as a Gaussian distribution [86]:

$$\mathcal{M}_i(l_t) = \mathbb{P}(l_t | p_t = e_i) = \frac{1}{\sigma\sqrt{2\pi}} e^{\frac{dist(e_i, l_t)^2}{2\sigma^2}}$$

where σ is the standard deviation of the positioning measurements. For example, when the input location observations are a sequence of GPS collected points, we use a standard deviation of 10 meters to estimate the noise distri-

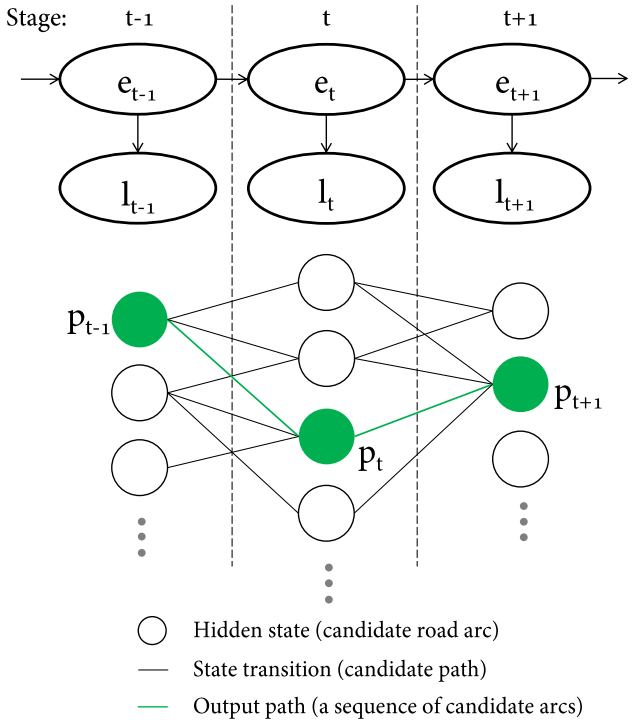


Figure 4.5: Illustration of state transition flow and Viterbi decoding algorithm.

bution [66]. $dist(e_i, l_t)$ represents the shortest distance from l_t to the candidate road arc e_i, which is the great circle distance on the surface of the earth between l_t and its corresponding match point m_t^i.

We also utilize the distance differences between the observation pairs and match point pairs to estimate the transition probabilities based on the study from Newson and Krumm [90]. Given two measurements l_{t-1}, l_t and their match points m_{t-1}^i, m_t^j, the transition probability of moving from e_i to e_j is:

$$\mathcal{T}_t^{ij} = \mathbb{P}(p_t = e_j | p_{t-1} = e_i) = \beta e^{-\beta \|d_l - d_m\|}$$

where d_l is the great circle distance between two location measurements and d_m is the shortest route distance from m_{t-1}^i to m_t^j.

Within a dynamic window size, this model is later decoded by our improved online algorithm and outputs $p_t = \{e_k, e_{k+1}, \ldots, e_i\}$, where $\{e_k, e_{k+1}, \ldots\}$ is the route path between e_{i-1} and e_i determined by the selected state transition path. This subset of candidate road arcs are generated as the most likely path for given observation l_t. It guarantees that the output paths are connected. In the following descriptions, we omit the $\{e_k, e_{k+1}, \ldots\}$ part in equations while we actually keep track of these connecting paths in the real system.

4.3.2 Improved Online Decoding

The aim of decoding is to discover the hidden state sequence that is most likely to have produced a given observation sequence. In the context of map matching, our algorithm needs to find the road arc sequence that is most likely to generate the collected location measurements. The traditional Viterbi decoder is a trellis algorithm (see Figure 4.5) defined as:

$$\delta_t(i) = \max_{p_1 p_2 \cdots p_{t-1}} \mathbb{P}\{p_1, p_2, \ldots, p_{t-1}, p_t = e_i, l_1, l_2, \ldots, l_t - 1|\lambda\}$$

which gives the highest probability that partial observation sequence and state sequence up to time step t can have, when the current state is i. The initialization and recursion step of the decoding phase are defined as:

$$\delta_1(i) = \pi_i \mathcal{M}_i(l_1)$$

$$\delta_t(j) = \max_{1 \le i \le N} [\delta_{t-1}(i) \mathcal{T}_t^{ij}] \mathcal{M}_j(l_t)$$

where N is the cardinality of candidate state set S, $S \subset E$. Usually the scale of the road network in modeling, $card(E)$, is relatively large, which leads to

inefficiency in decoding. Eddy narrows down the set of candidate states within S to accelerate the processing. We will elaborate on the details of downsizing later.

In each time step, we normalize the probability distribution to ensure $\sum_{j=1}^{N} \delta_t(j) = 1$. The backtracking pointer of the selected hidden state in each step is as follows:

$$\psi_t(j) = \arg\max_{1 \le i \le N} [\delta_{t-1}(i) \mathcal{T}_t^{ij}]$$

It terminates when the last observation is received and decoded by this procedure. The optimal path can be obtained by backtracking from the last matching result:

$$p_T = \arg\max_{1 \le i \le N} [\delta_T(i)]$$

$$p_t = \psi_{t+1}(p_{t+1})$$

However, this traditional type of decoder is not suited for real-time systems since the optimal state sequence cannot be computed until the entire input has been observed. Thus, some HMM-based frameworks have proposed several localizing strategies to fulfill the online output functionality. We first briefly summarize two widely used online decoding techniques and their limitations.

Fixed Segment/Sliding-Window

One simple and straightforward approach is to divide the trajectory into fixed-sized sequences and handle them independently. Given a desired latency D_d, the system simply fixes the segment size or window size as $\omega \le D_d$ and applies the Viterbi decoder to each segment/window to bound the maximum system delay.

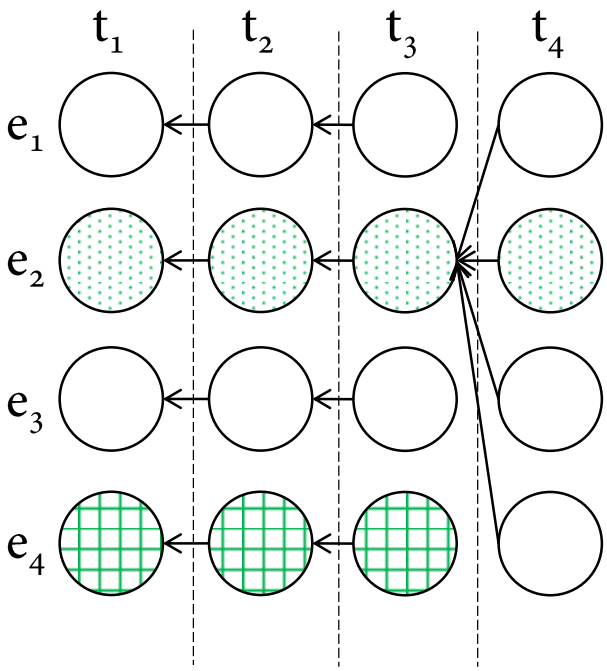


Figure 4.6: An example of online Viterbi decoding process.

For algorithms using a fixed segment (FS), the decoder waits for a segment-length of observations before decoding on the time slice from $t+1$ to $t+\omega-1$. In the sliding window method (FSW), the decoder considers only one new observation and moves the window forward one step a time. In the example illustrated in Figure 4.6, given $\omega = 2$, FS first reads measurements from t_1, t_2 in order, and generates the output path, say $P = \{e_4, e_4\}$. The next observation input of FS is from t_3. Thus, the matching output delays for the observations within the same segment are different. FS outputs the result for l_1 with 1 time step delay (to wait l_2 to fill the segment size) while the matching delay of l_2 is 0 (we do not add the client-server transmission time and matching processing time into the delay calculation since we focus on the decoding delay in this study). Differently, given $\omega = 2$, FSW takes l_3 as input right after generate the matching result of l_1, by sliding the window from $[t_1, t_2]$ to $[t_2, t_3]$. The matching

delay for all observations from this method is constant (except for the last ω location measurements since there is no future room to slide forward).

Usually, a larger window size leads to a more accurate matching result but a longer output delay, and vice versa [79, 133, 111]. In the previous example, given $\omega = 3$, the FS/FSW decoder can generate an arbitrary path for segment/window $[t_1, t_3]$. One plausible path could be $P' = \{e_4, e_4, e_4\}$, illustrated as a lattice-pattern circle sequence. However, when the decoder receives l_4 , it would have enough knowledge to recognize that P' is not a possible output. Although this example is only an undesirable case and may not be triggered frequently in real scenarios, it illustrates the tradeoff we need to carefully deal with between the accuracy and latency. This accuracy degradation occurring from the sub-optimal path generation may arise due to certain pre-defined segment- or window-size settings. Therefore, a decoding algorithm which can intelligently choose a dynamic window size is preferred.

Convergence State Discovery

Some HMM-based applications adopted another technique named Convergence State Discovery (CSD, also called fusion point finding), which is capable of finding the optimal path before the entire trajectory is received [14, 37]. The basic idea in this algorithm is to delay the label generation until encountering a converging state like e_2 at t_3 in Figure 4.6. When CSD reads the input observation l_4 and calculates related probabilities, it sees that all backtracking pointers point to the same state, e_2. It is easy to prove that all future surviving paths will contain the same sub-path before this convergence state. Thus at time t_4, CSD can output the matching result for observations l_1, l_2 and l_3, $P = \{e_2, e_2, e_2\}$ (the dot-filled circle chain).

This algorithm has the advantage of holding the promise that the generated output path is identical to the result from the original Viterbi decoder. However, a serious issue that this algorithm may encounter is the absence of a fusion point in some real problems or some pre-defined probability normalization rules. The matching delay is prolonged if the convergence state comes late, and may persist to the end of the observation sequence if no such point exists. In other words, CSD is not delay-bounded and in the worst case degenerates to the original Viterbi algorithm. Therefore, for most latency-sensitive applications, this decoding algorithm is unfit.

Improved Online Decoder based on Ski-rental Model

To better interpret the tradeoff between the map matching accuracy and latency, we model the online decoding phase as a ski-rental problem in this study. The ski-rental model, also known as "rent or buy" dilemma, is one of the fundamental problems in online algorithms. This problem was first abstracted by Karlin et al. and used in a communication minimization algorithm [55]. In a classic ski rental problem, a skier may rent skis for R per day or buy them for B dollars. At the end of any day, the skier may break his legs along with the skis, or in some other way irrevocably finish skiing. The goal is to develop an online strategy minimizing the cost spent on skiing, where the cost is compared to the cost of an optimal offline strategy for the same input. The worst-case ratio between these two amounts is called *competitive ratio*.

Inspired by one of its variants, "Multislope Ski Rental" [78], we use a generalized model with a inconstant buying price B_t that changes over time in

our case. Obviously, the total cost of skiing is

$$\mathcal{C}_s = B_{\hat{t}} + R \times \hat{t} \tag{4.4}$$

where the skier decides to buy the skis in the evening of the $\hat{t}^{th}$ day.

Similarly, in our scenario, we model the accuracy penalty and latency penalty as the buying price and rental rate, respectively. We need to decide whether to stay in the current decoding state and pay a certain amount of latency cost per time unit, or output the present matching result and pay some large accuracy penalties but with no further delay penalty. Without loss of generality, we assume the location observation l_0 measured at t_0 has been matched to the road network and l_1 from t_1 is under the decoding phase currently. The future information up to $\hat{t}$ is observed and transferred to the decoding system to help the joint probability computation. Moreover, the decoder decides to output the matched result p_1 at time $\hat{t}$. Straightforwardly, the delay of decoding l_1 is $\hat{t} - t_1$, which is similar to the rental rate that a skier has to pay before a buying decision. Meanwhile, to better estimate the accuracy of the matching roads, we leverage the probability distribution $\delta_{t_1,\hat{t}}(j)$ which indicates the likelihood of each state e_j being the matching road. Notably, this is different from $\delta_{t_1}(j)$ since the system involves future information into the inference chain. We first calculate $\delta_{t_1}(j)$ considering that the matching result p_0 for the observation l_0 has been generated already,

$$\delta_{t_1}(j) = \max_{1 \le i \le N} [\delta_{t_0}(i) \mathcal{T}_{t_1}^{ij}] \mathcal{M}_j(l_1)$$

$$\delta_{t_0}(i) = \begin{cases} 1, & \text{if } p_0 \in e_i. \\ 0, & \text{otherwise.} \end{cases}$$

where the distribution of δ_{t_0} is determined. With all the future observations we waited and received from t_1 to $\hat{t}$, we afterwards obtain,

$$\delta_{t_1,\hat{t}}(j) = \sum_{i=1}^{N} \delta_{\hat{t}}(i)$$

$$\text{if} \quad \psi_{t_1,\hat{t}}(i) = j$$

where $\psi_{t_1,\hat{t}}(i)$ is the backtracking function from time frame $\hat{t}$ to t_1

$$\psi_{t_1,\hat{t}}(i) = \psi_{t_1}(\psi_{t_2}(...\psi_{\hat{t}-2}(\psi_{\hat{t}-1}(i))))$$

Thus, $\delta_{t_1,\hat{t}}(j)$ is the sum of $\delta_{\hat{t}}(i)$ where e_j at time step t_1 and e_i at time step $\hat{t}$ are on the same candidate path connected by standard Viterbi backtracking pointers. As illustrated in Figure 4.7, the probability that e_1 is the output matching result is the sum of $\delta_{t_4}(e_1)$, $\delta_{t_4}(e_2)$ and $\delta_{t_4}(e_3)$ when computing at time t_4. For each $e_j \in S$, $\delta_{t_1,\hat{t}}(j)$ presents the probability that l_1 should be matched to e_j after future observations up to $\hat{t}$ are considered into the HMM framework.

Intuitively, if only one state is calculated with a significantly high probability and the other states' likelihoods are near zero, we can deduce confidently that this state is the matching road and generate this road arc as the output label. To better describe the distribution characteristics and incorporate this into our decoding procedure, we use the information entropy of $\delta_{t_1,\hat{t}}(j)$ as a proxy of the accuracy penalty.

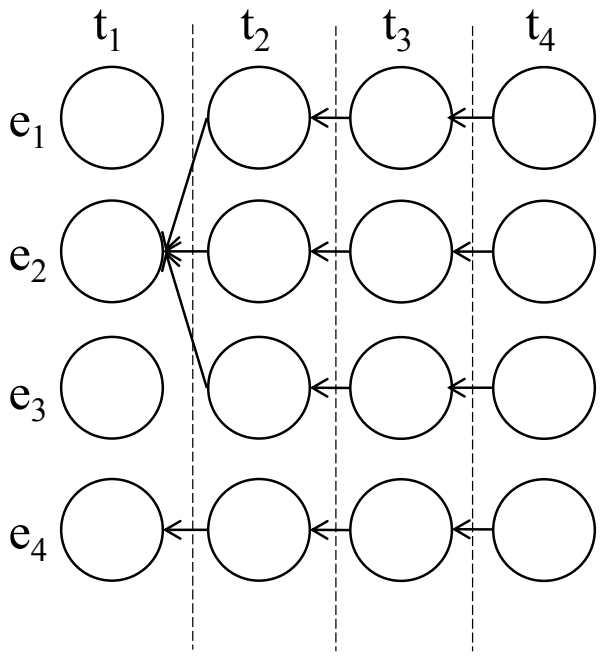


Figure 4.7: Illustration of the state probability recalculation after future location observations are received.

$$\mathcal{H}(t_1, \hat{t}) = -\sum_{j=1}^{N} \delta_{t_1,\hat{t}}(j)\ log\ \delta_{t_1,\hat{t}}(j)$$

The entropy $\mathcal{H}(t_1, \hat{t})$ is a logarithmic measure of the number of states with significant probability of being occupied, which indicates the degree of *uncertainty* at time step t_1 after receiving future observations up to $\hat{t}$. According to the definition of entropy function, the larger the value $\mathcal{H}$ is, the higher the uncertainty of this outcome state could be. The highest entropy outcome is achieved when $\delta_{t_1,\hat{t}}(j)$ is evenly distributed among all candidate states. On the other hand, if $\mathcal{H}$ is close enough to zero, it means that one state is extremely outstanding within the candidate space. This plays the same role as the buying price B_t in the ski-rental model. Therefore, in accordance with Equation (4.4), we derive

our objective cost function as the sum of the accuracy and delay penalties,

$$\mathcal{C}(t_1, \hat{t}) = \mathcal{H}(t_1, \hat{t}) + \gamma(\hat{t} - t_1)$$

where γ is the parameter to control the tradeoff between accuracy gain and delay cost. If the real-time system is extremely sensitive to the latency, a larger value of γ should be chosen. By contrast, if the monetary cost of false road matching is expensive, a small γ should be considered to penalize more on the accuracy part.

Similar to the ski-rental model, whose ultimate target is to determine the buying date, here we need to provide a strategy to decide at which $\hat{t}$ we should stop delaying and output the matching result $\arg\max_j[\delta_{t_1,\hat{t}}(j)]$. Thus, our online system needs to choose an appropriate label generation time $\hat{t}$ to minimize the cost $\mathcal{C}$.

Clearly, the delay cost accumulates linearly like a monotonically increasing function. If the accuracy penalty $\mathcal{H}(t_1, \hat{t})$ changes arbitrarily over time, its sum $\mathcal{C}$ is difficult to be minimized. Thus, here we assume that given t_1, $\mathcal{H}$ is a monotonically decreasing function of variable $\hat{t}$. The physical meaning of this assumption is that we believe the uncertainty of the state outcome at a certain time step would decrease as a growing number of future observations are analysed within the decoding procedure. We will show in experimental section that our assumption is reasonable across the entire test dataset.

Thereby, we need to minimize the sum of a decreasing function and an increasing function. In the ski-rental model, the *break-even* algorithm is known as the best deterministic algorithm for this set of problems [55]. We adopt a similar idea and choose the time point $\hat{t}$ when $\mathcal{H}(t_1, \hat{t})$ is equal or less than the

value of $\gamma(\hat{t} - t_1)$, to output the matching road result. The intuition behind this algorithm is to adaptively adjust the window size based on the uncertainty of the state matching. If the uncertainty degree is high, the algorithm should extend the window size to absorb more future location observations before generating the road arc label. Conversely, if the initial $\mathcal{H}$ value is low enough or the function $\mathcal{H}$ drops rapidly, the window should become smaller and the matching output will be generated soon.

The pseudo-code for general cases is detailed in Algorithm 1. The "+" operator on line 10 means to attach a new output to the global sequence. P and T' can be implemented as a *pipe* with capacity of 1, so that once a new output p_i is generated, it can be consumed by an upstream real-time application immediately, and the latency is exactly $t'_i - t_i$.

Accuracy and Latency Analysis

To better illustrate the advantage of our improved online decoding algorithm, here we present a theoretical competitive and upper-bound analysis for accuracy and latency, respectively. First we provide the competitive ratio of our decoder, which is the worst-case ratio between the cost of the solution found by our algorithm and the cost introduced by an optimal solution. Assume for a given l_i received at t_i, Eddy generates the according road arc label at time t. Two situations need to be considered when analyzing the worst case — one is that the actual optimal output time step T_o is earlier than t, and the other is $T_o > t$. The cost of the optimal solution is $\mathcal{H}(t_i, T_o) + \gamma(T_o - t_i)$. If $T_o < t$, it indicates that, even with more measurements adopted, the cost decrease from the accuracy penalty $\mathcal{H}$ does not make up for the cost increase caused by the latency penalty. In other words, the concentration expectation of the state distribution based

on future observations is not achieved. The worst case in this situation is that $\mathcal{H}(t_i, t_i) = \mathcal{H}(t_i, t) + \epsilon$ where ϵ is a real number approaching zero (it cannot be zero since $\mathcal{H}$ is a monotonically decreasing function), and the optimal output is $T_o = t_i$. The optimal solution outputs the map matching result immediately since the future observations benefit nothing to the decoding process in order to involve no latency penalty to the cost function,

$$\mathcal{C}(t_i, T_o) = \mathcal{C}(t_i, t_i) = \mathcal{H}(t_i, t_i)$$

Since our algorithm generates a road arc result at t, not $t-1$, we have

$$\mathcal{H}(t_i, t) < \gamma(t - t_i)$$

$$\mathcal{H}(t_i, t-1) > \gamma(t - 1 - t_i)$$

Also, $\mathcal{H}(t_i, t) - \mathcal{H}(t_i, t-1) < \epsilon < \gamma$, so we obtain

$$\gamma(t - 1 - t_i) < \mathcal{H}(t_i, t) < \mathcal{H}(t_i, t_i)$$

Thus, the cost of our method is,

$$\begin{aligned}
\mathcal{C}(t_i, t) &= \mathcal{H}(t_i, t) + \gamma(t - t_i) \\
&= \mathcal{H}(t_i, t_i) + \gamma(t - 1 - t_i) + \epsilon + \gamma \\
&< \mathcal{C}(t_i, T_o) + \mathcal{C}(t_i, T_o) + \epsilon + \gamma \\
&= 2\mathcal{C}(t_i, T_o) + \epsilon + \gamma
\end{aligned}$$

If $T_o > t$, the worst case is that $T_o = t + 1$ and $\mathcal{H}(t_i, T_o) = 0$ because this is the lowest value pair for both two penalties and all other cases would achieve

a higher $\mathcal{C}(t_i, T_o)$. Thus the cost of the optimal solution is,

$$\begin{aligned}\mathcal{C}(t_i, T_o) &= \mathcal{C}(t_i, t+1) \\ &= \mathcal{H}(t_i, t+1) + \gamma(t+1-t_i) \\ &= 0 + \gamma(t-t_i) + \gamma \\ &> \frac{\gamma(t-t_i) + \mathcal{H}(t_i, t)}{2} + \gamma \\ &> \frac{\mathcal{C}(t_i, t)}{2}\end{aligned}$$

Thereby, we proved that the cost of our algorithm $\mathcal{C}(t_i, t)$ is no more than 2 times of the cost introduced by all the other solutions plus a constant, and thus our improved online decoder is a 2-competitive algorithm.

Next, we illustrate that our improved online decoding is latency-bounded. Assume at time t, the algorithm has not generated the road arc output for a given measurement l_i. Since we adopt the break-even condition, we have $\mathcal{H}(t_i, t) > \gamma(t - t_i)$. In addition, $\mathcal{H}$ is a monotonically decreasing function and of course $t > t_i$ because we cannot perform map matching without receiving the measurement. Thus, we have $\mathcal{H}(t_i, t) < \mathcal{H}(t_i, t_i)$. Clearly, by the transitive property of inequalities, we obtain $\gamma(t - t_i) < \mathcal{H}(t_i, t_i)$. Therefore, the upper-bound of map matching delay of l_i is $\mathcal{H}(t_i, t_i)/\gamma + t_i$, which is only determined by the characteristic of distribution δ_{t_i}. The matching process of every incoming observation would terminate for sure even if the entire measurement input is infinite.

Candidate State Space Reduction

To make the decoding process more efficient, we narrow the range of candidate states $card(S)$ in our HMM model. Due to the fact that the vehicles usually

drive at a limited speed during the time interval between two consecutive sample measuring locations, the current location measurement (except the first one) should not be too far away from the previous one. It is very likely that all candidate road arcs of the current location observation fall into a small area around the previous sample point. Therefore, we employ the radial search method proposed by Fang and Zimmermann, to find the candidate road arcs of a location measurement point instead of using the traditional range query [31]. It utilizes the topological information of the road network to radially check each candidate road arc in the vicinity., while employing the speed constraints of previous road arcs to limit the search scope.

4.4 Experiments

4.4.1 Evaluation on Pedestrians Attached Sensors

We implemented both of our location data correction algorithms, and evaluated them on a set of sensor-annotated videos publicly available at the Geovid website. We first report the accuracy enhancement we achieved for location measurements along with those video clips. We selected first batch of sensor-annotated videos and their sensor dataset retrieved from the Geovid website, using their provided APIs. Among these 87 videos, we perform experiments on 63 selected ones, since the other 24 videos contained only a few GPS samples and had a relatively small recording duration (typically less than one minute). The smartphones used in data collection range from various Apple iPhone devices to a number of Android devices (Motorola droid, Samsung Galaxy S, ASUS Transformer, HTC Desire). We illustrate the GPS data correction results of

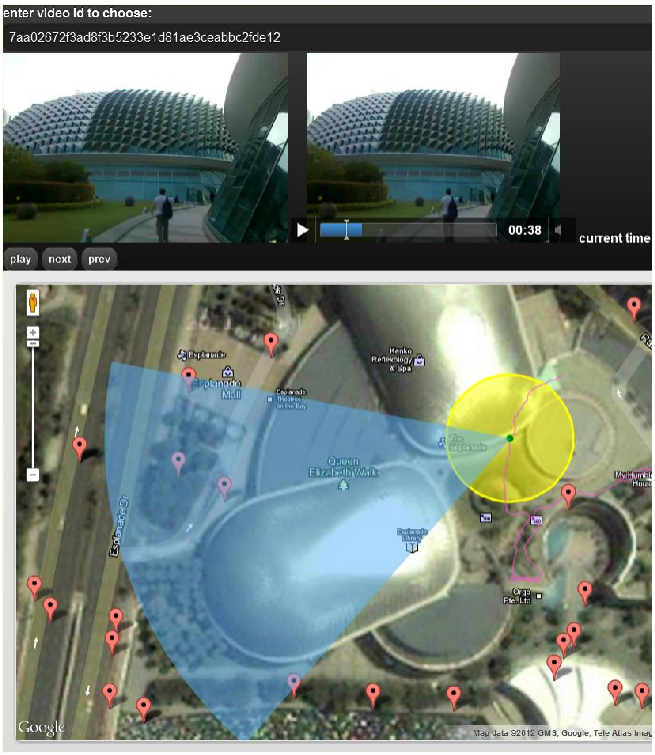
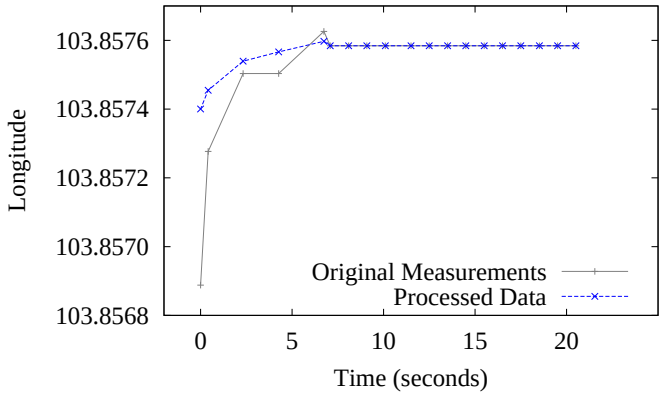


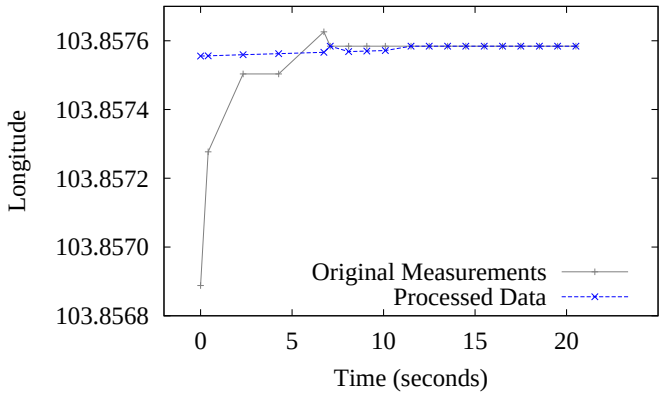
Figure 4.8: A screenshot of our GPS annotation tool. It allows users to manually modify an incorrect GPS location to a better-matching place. To assist better estimating correct locations, it provides interactive frame-by-frame image navigation, video playback, and current compass direction (depicted as pie-slice). Yellow-colored circle area represents the accuracy range of an original GPS sample.

one data segment and 63 GPS sequence data sequences by both algorithms we proposed. In this experiment, we set function $g(a_i) = a_i^2$ in terms of the physical meanings of both standard deviation in the normal distribution and accuracy measurements in GPS generated data. We use the threshold $TH_a = 40$ meters.

To establish the ground-truth dataset of individual GPS samples, we developed a web-based in-house utility (shown in Figure 4.8). It displays a video frame at a specified time instant along with its corresponding GPS location on the map. It also allows a user to freely select a specific location on the map,



(a) Kalman filtering based algorithm.

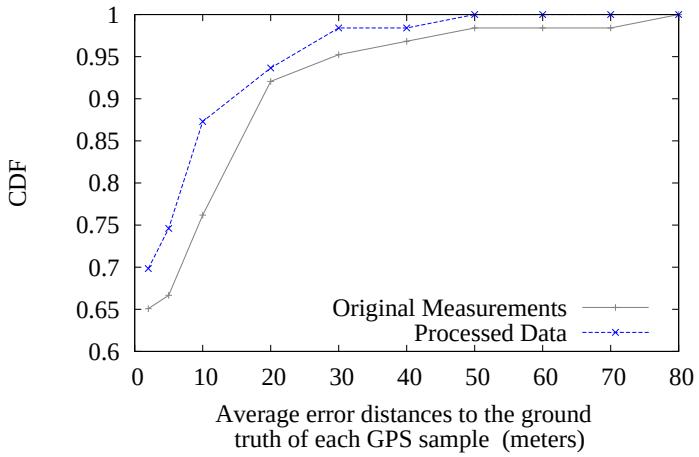


(b) Weighted linear least squares regression based algorithm.

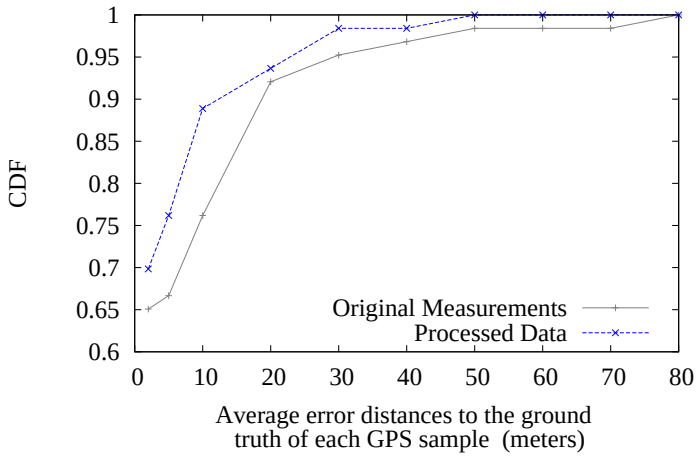
Figure 4.9: Corrected longitude value results of one GPS data segment.

where the user believes to be the correct location. The newly chosen location was then automatically stored in our database system. To assist the users to easily figure out the right location of the GPS sample, we also displayed the accuracy range of the sample on the same map interface, using a yellow-colored circle centered at its reported location. Using this utility, we collected 1679 annotated ground truth samples out of 10069 raw GPS samples for all 63 videos.

First we illustrate how our algorithms work for one piece of GPS sequence data segment. We apply our approaches to a segment located in the beginning 20 seconds of a GPS data file, which contains the typical error pattern we



(a) Kalman filtering based algorithm.



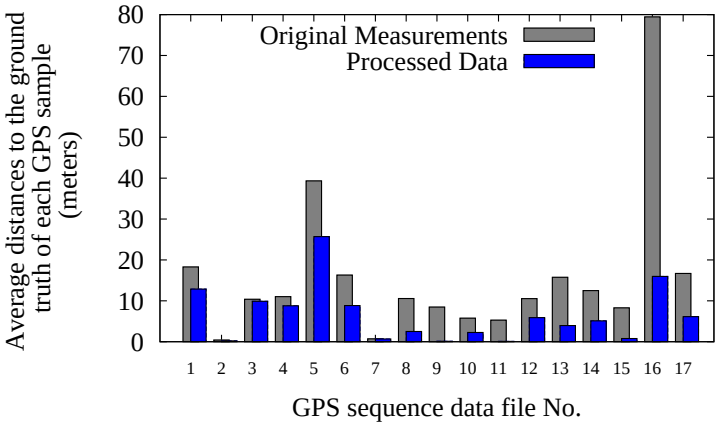
(b) Weighted linear least squares regression based algorithm.

Figure 4.10: Cumulative distribution function of average error distances.

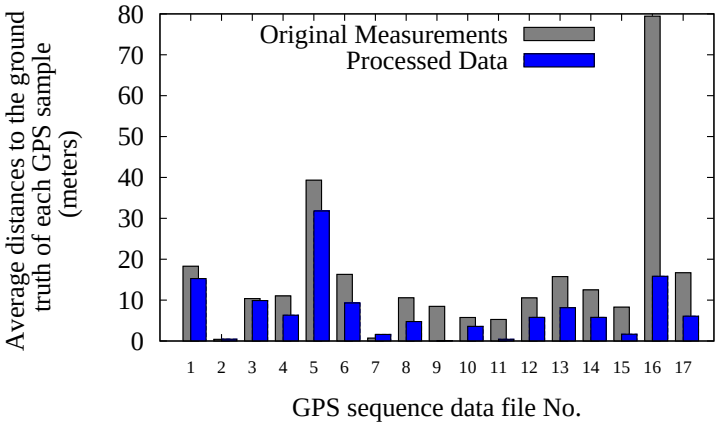
mentioned earlier (See Figure 4.9. The height of each point represents the total amount of GPS sequence data files whose average distance to the ground truth positions is less then the given distance value.). As we can see, the origin GPS longitude data before 6 seconds are very spotty and unreasonable, which cause the jumping phenomenon on a map interface. After the correction phase by our two algorithms, we find the processed GPS data approach the reasonable longitude values by a distinct improvement.

Afterwards, to quantitatively evaluate two proposed algorithms, we compute the average distance between every processed sample and its corresponding ground truth position for each GPS sequence data file, and compare these values to the average distance between every measurement sample and the ground truth position. By comparing these two average error distances, we report the correctness of processed data. On average, the Kalman filtering based algorithm and the weighted linear least squares regression based algorithm improve the GPS data correctness by 16.3% and 21.76%, respectively. Figure 4.10 illustrates a Cumulative Distribution Function (CDF) for both algorithms. We increase the proportion of GPS data with low average error distance and shorten the largest sequence average error distance by around 30 meters (the line of processed data meet $y = 1$ at $x = 50$, while the line of original measurements achieve at $x = 80$).

Moreover, we apply our algorithms to 17 highly inaccurate datasets (i.e., the highest accuracy value $max(a_i) > 50$ meters). We found our algorithms reduce the average distances between the measured positions and the ground truth data to a great extent. The Kalman filtering based algorithm and the weighted linear least squares regression based algorithm reduce the average error distances by 39.82% and 48.18%, respectively. The reason that the regression model performs better here is the less assumptions it makes on the trajectory. As we mentioned earlier, regression model assumes that each segment only contains linear movement, while the Kalman filtering model treats each segment as a constant velocity (a stronger assumption). Figure 4.11 illustrates the average error distance reductions of every GPS data sequence file. For some extremely inaccurate sequences like file 16, we significantly reduce the error distance from near 80 meters to less than 20 meters.



(a) Kalman filtering based algorithm.



(b) Weighted linear least squares regression based algorithm.

Figure 4.11: Average error distance results between the corrected data and the ground truth positions of highly inaccurate GPS sequence data files.

4.4.2 Evaluation on Vehicle Attached Sensors

To evaluate our Eddy system, we implemented the other two online Viterbi decoding strategies, FS and FSW, as comparisons. As previously described, the CSD strategy always generates the optimal solution (identical to the offline decoder's result) but does not guarantee any delay upper-bound, which usually involves a long latency (in the order of minutes) and is not applicable to real-time services [37]. Thus we did not compare our algorithm with CSD in this

study.

In our experiments, we adopt the public real-world dataset collected in Seattle provided by Newson and Krumm [90], including the relevant road network, GPS trajectory data, and ground truth. The road network comprises more than 150,000 road arcs. The raw GPS trajectory data is a 50-mile route in Seattle which is sampled at 1 Hz and took about 2 hours to drive, giving 7,531 time-stamped latitude/longitude pairs. The ground truth contains a sequence of road arcs with the directions in which the vehicle actually travelled. Since it is impossible for us to know the exact actual location of the vehicle in the road network corresponding to each GPS sample point, only the path taken by the vehicle is viewed as the ground truth. We also adopt the underlying HMM model parameters, σ and β, which have been tested and verified in their study.

We focus on two evaluation aspects, accuracy and latency, in these experiments. First, we compute the actual trends of information entropy $\mathcal{H}$ for all the location measurement points from the dataset. We show that our assumption is reasonable that it is a monotonically decreasing function of variable $\hat{t}$. Afterwards, we apply our method and two baseline algorithms to the whole dataset to compute and visualize the tradeoff between accuracy and latency.

Our improved online decoding algorithm and the other two comparison methods are all implemented in C# and connected with a lightweight in-memory database, SQLite. Since we focus on the road arc label generation delay instead of thye real processing time, this database is completely stored and processed in RAM.

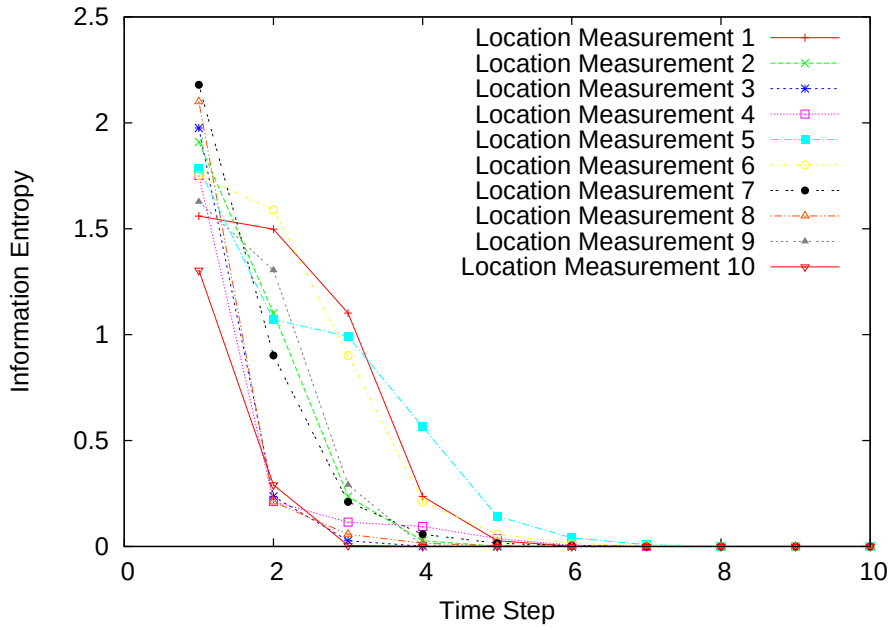


Figure 4.12: Information entropy trends of 10 example location measurements.

Accuracy Penalty Trend

We utilize the radial search method to reduce the candidate set of road arcs, and we set the candidate state size parameter $\alpha = 1.8$ in our experiments, which has been empirically tested earlier [31]. This leads to the property that only a small set of candidate states e_i share the matching probability and thereby the distribution concentrates more quickly than in the case where we use the whole road network as the candidate set. We calculate the information entropy, which is considered as the accuracy penalty proxy in our algorithm, for every location measurement in the scope of the whole trip. For each measurement l_i, we record and update its entropy value changes when future observations $l_{i+1}, l_{i+2}, \ldots, l_n$ are received.

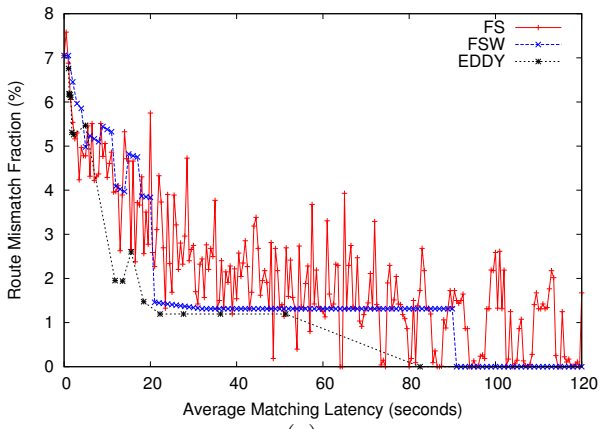
Figure 4.12 illustrates 10 example trends of the location measurement's information entropy as the time elapses (one new observation received at every time step). As shown, the value of entropy function $\mathcal{H}$ is relatively high when

only the current measurement is received and no future observation is incorporated into the model. It indicates the difficulty of generating the matching result immediately. As the time step increases, $\mathcal{H}$ turns to be a monotonically decreasing function as we hypothesized.

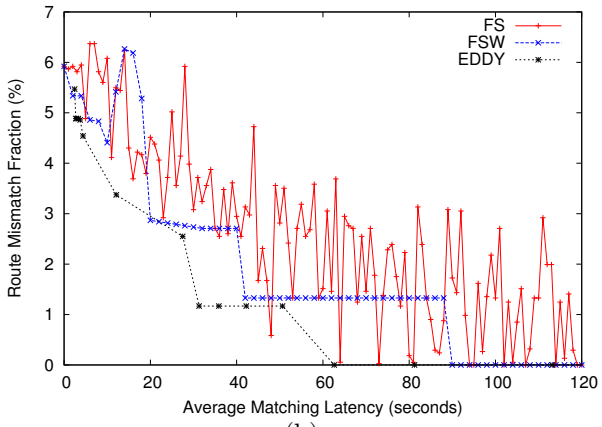
If the value of $\mathcal{H}$ increases as the time step moves forwards for a given l_i, we judge that this entropy function is not a monotonically decreasing function, and we also record the time step where the entropy value increases as the increasing point. Among the entire trajectory dataset, we find that 91.53% of the measurements' entropy function is monotonically decreasing. Moreover, in the remaining part of this dataset, 5.52% of functions' increasing point appears after receiving more than 400 future observations. It is very likely that the system has already passed the break-even point before seeing such a large number of future observations. In other words, 97.05% of functions are actually decreasing if the delay of a system is limited to less than 400 seconds, which is a reasonable setting in the context of a real-time system. Additionally, if the real-time system only considers future observations within the range of 50 samples, 100% of $\mathcal{H}$ satisfies our assumption. This result intuitively makes sense because of the underlying logic in that the more future observations are incorporated into the decoding model, the more confidently we can determine which road the vehicle is driving on.

Error and Delay

To illustrate the tradeoff between the matching accuracy and latency, we apply our system and two comparison algorithms to the Seattle trajectory dataset with different γ values and window sizes w. Different sampling periods are considered in our experiments as well to show the robustness of our algorithm



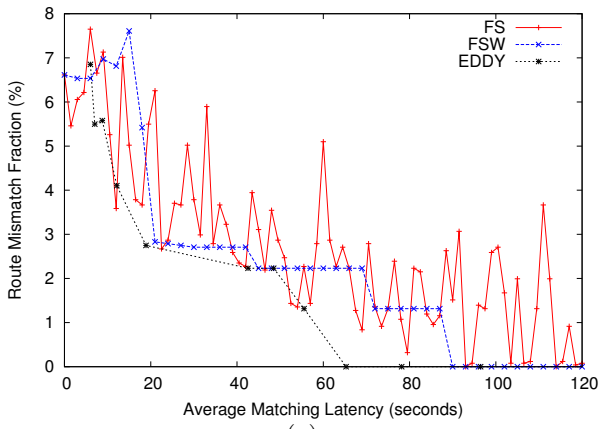
(a)



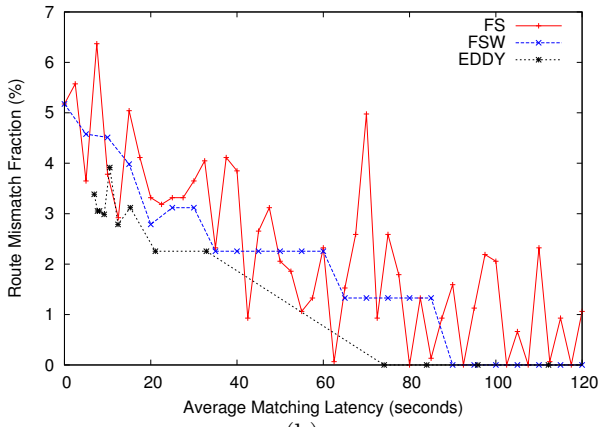
(b)

Figure 4.13: The accuracy (in RMF) and latency (in seconds) of map matching results on different location measurement sampling intervals: 1 observation sample (a) per second, and (b) every 2 seconds.

under different location measuring rates. We adjust the γ value from 0.01 to 2 to tune the tradeoff between the road arc mismatch rate and delay time. The parameter w varies according to the change of the location measurement sampling intervals. For example, in order to obtain an accuracy change from no delay at all to a latency of 120 seconds, we tune the w value from 0 to 120 for FSW, and from 1 to 241 for FS, with a sampling period of 1 second. The reason is that FS generates labels for all location observations within the current window at once (when the window is full), so that the location mea-



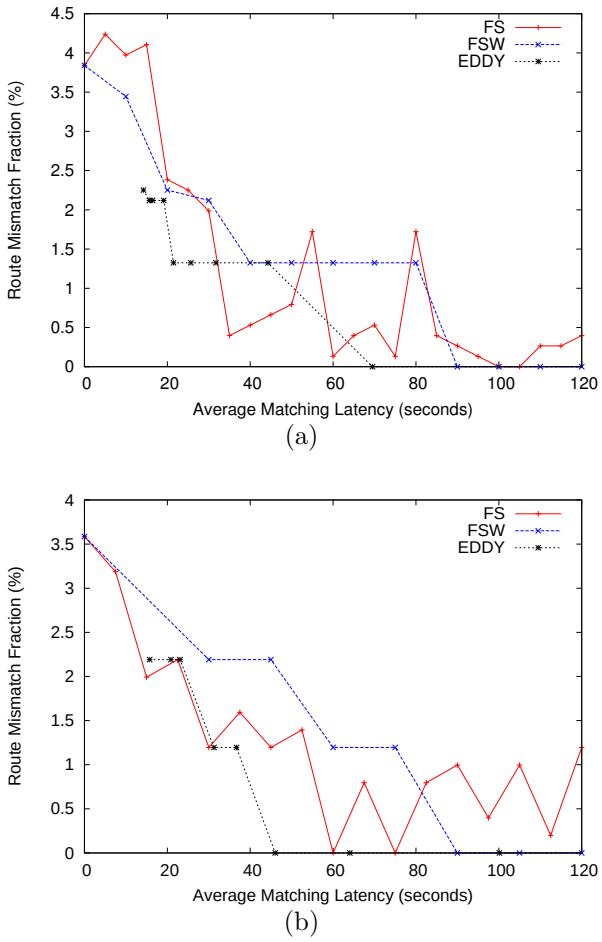
(a)



(b)

Figure 4.14: The accuracy (in RMF) and latency (in seconds) of map matching results on different location measurement sampling intervals: 1 observation sample (a) every 3 seconds, and (b) every 5 seconds.

surements in the second half of the window have lower *effective latency* than the measurements in the first half. Clearly, the road arc label of the last location observation tucked into the window will be matched and generated by FS immediately without any latency no matter how large the window size is. Thus, we consider the average effective latency among the observations within the same window, $(w-1)/2 * (\text{sampling period})$, as the average latency. Similarly, when the sampling period becomes 10 seconds, we evaluate the w value from 0 to 12 for FSW, and from 1 to 25 for FS, respectively, to compute the



(a)

(b)

Figure 4.15: The accuracy (in RMF) and latency (in seconds) of map matching results on different location measurement sampling intervals: 1 observation sample (a) every 10 seconds, and (b) every 15 seconds.

mismatch percentage trend from no delay to a latency of 120 seconds.

The matching accuracy is measured by the Route Mismatch Fraction (RMF). This fraction is the total length of a false positive route in P and a false negative route in G_e divided by the length of the original route. We report RMF in percentage for each experiment and a higher RMF result indicates more erroneous road arcs are generated by the online map matching algorithm.

As illustrated in Figure 4.13 to 4.15, we report the map matching accuracy trend from immediate label generation to a latency of 120 seconds, under

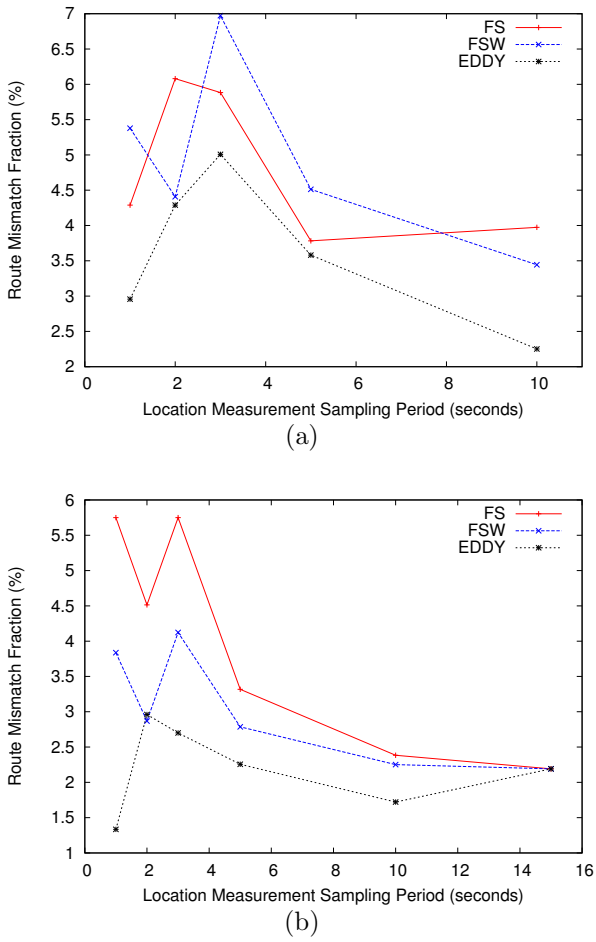


Figure 4.16: The comparisons of map matching results' accuracy for different location measurement sampling intervals under fixed latency constraints of: (a) 10 seconds and (b) 15 seconds.

different measurement sampling periods. First, all figures show an overall declining trend of road arc mismatch fraction, which is sensible in that less error results are generated if more future location observations are analyzed within the HMM model. Second, the output quality of the FS algorithm is much less stable than the other two. Although the general trend of FS is descending as well, more fluctuations arise when the latency increases. By contrast, FSW and our algorithm are more stable, which means the matching results are confidently expected to be more accurate if more future information is provided.

Most importantly, it is also observable that the curve of Eddy is mostly below FS and FSW. It indicates that our Eddy map matching system outputs better results in most cases with respect to two aspects: a) under the same latency constraints, the RMFs of Eddy are mostly the lowest one, especially for high sampling rate location dataset. In the experiment using 1-second sampling rate GPS data with latency constraint of 15 seconds , we decrease the route mismatching fraction from 5.7%(the result of FS) down to 1.3%. and b) under the same accuracy constraints, Eddy is able to achieve the shortest latencies. In the experiment using 1-second sampling rate GPS data with accuracy demand of 2% mismatching fraction, we shorten the latency from more than 110 seconds(the result of FS) down to less than 20 seconds.

Figure 4.16 illustrates the online map matching accuracy improvements under the same latency constraints, 10 seconds and 20 seconds respectively. As shown, when applying our algorithm to the location measurement datasets with different sampling rates, our matching result almost always outperforms the other two methods with less error erroneous generations.

Moreover, we also notice from the experiments that the RMF value of Eddy stably reaches 0 much earlier than FS and FSW (under different sampling rates shown in Figure 4.13 to 4.15). It means that our system is able to achieve a stable 100% accuracy of the road arc generation results with a much shorter latency.

4.5 Summary

In this chapter we presented two frameworks for pedestrian-attached and vehicle-attached location sensor data. First we analyzed several typical error pat-

terns for real-world pedestrian-attached positioning data, and proposed two approaches to improve the location measurement accuracy while relying purely on the GPS generated data. The experimental results show that our methods are highly effective in enhancing accuracy by up to 48%. For vehicle-attached trajectories, we presented a real-time HMM-based map matching system, Eddy, based on an improved online Viterbi decoding algorithm. Our method analyzes the tradeoff between the map matching accuracy and latency, and incorporates a ski-rental model and its best-known deterministic algorithm to solve the online decoding problem. Therefore, our system is capable of dynamically selecting the window size according to characteristics of the candidate state probability distribution. In our future work we plan to explore the possibility of involving nondeterministic algorithms into the online decoding phase to yield a better map matching accuracy and a shorter label generation delay. We believe that such processed, highly accurate location sensor data are useful for other sensor-aided mobile media applications.

Input: A location trajectory $L = \{l_1, l_2, \ldots, l_n\}$, and its according time sequence $T = \{t_1, t_2, \ldots, t_n\}$, both of which could be infinite. A set of candidate road arcs (hidden states) $E = \{e_1, e_2, \ldots, e_N\}$.

Output: A sequence of path $P = \{p_1, p_2, \ldots, p_n\}$, where $P \subset E$, and each p_i's mapping output time $T' = \{t'_1, t'_2, \ldots, t'_n\}$.

$P \leftarrow \{\emptyset\}, T' \leftarrow \{\emptyset\}$
$\hat{t} \leftarrow t_1$
foreach $t_i \in T$ **do**
  if $t_i \geq \hat{t}$ **then**
    $\hat{t} \leftarrow \hat{t} + 1$
  end
  while $t_i < \hat{t}$ **do**
    if $\mathcal{H}(t_i, \hat{t}) \leq \gamma(\hat{t} - t_i)$ **then**
      $t'_i \leftarrow \hat{t}$
      $p_i \leftarrow \arg\max_{1 \leq j \leq N}[\delta_{t_i, t'_i}(j)]$
      $P \leftarrow P + p_i$, $T' \leftarrow T' + t'_i$
      $t_k \leftarrow t_i + 1$
      while $t_k \leq \hat{t}$ **do**
        foreach $e_i \in E$ **do**
          update $\delta_{t_k}(e_i)$
          update $\phi_{t_k}(e_i)$
        end
      end
      leave loop
    end
    else
      $\hat{t} \leftarrow \hat{t} + 1$
      foreach $e_i \in E$ **do**
        update $\delta_{t_i, \hat{t}}(e_i)$
      end
    end
  end
end

Algorithm 1: Improved Online Viterbi Decoding

CHAPTER 5

Orientation Sensor Data Accuracy Enhancement

5.1 Introduction

In addition to the location accuracy we have discussed in Chapter 4, orientation is another type of sensor data that is increasingly used in many application and also has the necessity to enhance its accuracy. This chapter introduces a novel hybrid framework which corrects orientation data measured in conjunction with mobile videos based on geospatial scene analysis and image processing techniques. In particular, our system collects visual landmark information and matches it against GIS data sources to infer a target landmark's real geolocation. By knowing the geographic coordinates of the captured landmark and the camera, we are able to calculate corrected orientation data. While we

describe our method in the context of video, images can be considered as a specific frame of a video, and our correction approach can be applied there as well. Our contributions in this work include:

- Design and prototype implementation of algorithms to effectively enhance the accuracy of noisy camera orientation data.
- Accuracy evaluation between the corrected results and the raw (uncorrected) data, and performance comparison between our framework and existing state-of-the-art methods.

In most modern mobile operating systems, the camera orientation θ is measured and presented by how many degrees a northward unit vector needs to rotate to this vector clockwise in 2D geospace, *i.e.*, $\theta \in [0, 360)$. For example, if the camera is facing due east, then $\theta = 90$, and if $\theta = 180$, the camera is shooting southward. The tilting operation of the camera is not covered in this study. We plan to further elaborate on 6 degrees of freedom (DOF) camera pose correction techniques in 3D geospace as part of our future work. The phrases *camera orientation* and *viewing direction* are used interchangeably in our study.

5.2 Orientation Data Correction

Figure 5.1 illustrates the overall process flow and how our proposed orientation data correction module fits in. In the architecture, geospatial sensor data is collected during the video recording on a mobile platform and uploaded to a NoSQL database in the server side. Afterwards, most sensor-aided applications would directly utilize the raw sensor data to guide the multimedia content anal-

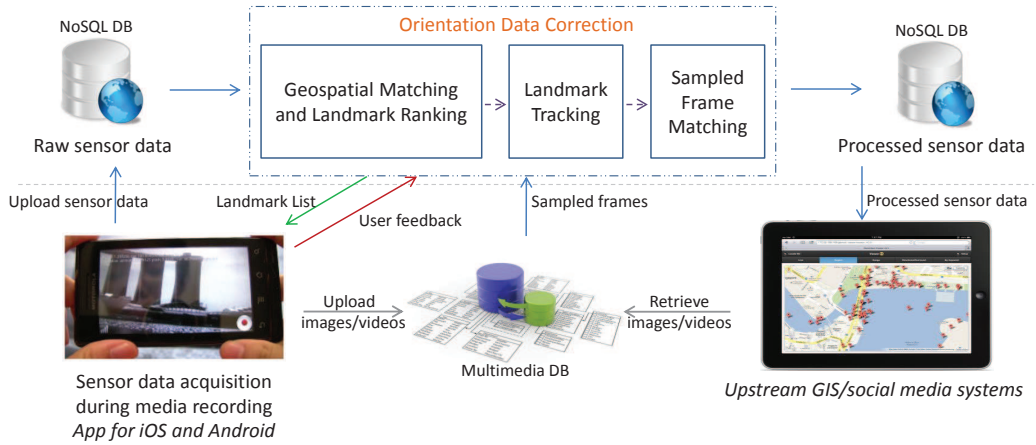


Figure 5.1: The overall architecture and the process flow of the proposed framework. Raw orientation sensor data is enhanced to provide more accurate directional information to upstream applications.

ysis and management as we exemplified in Chapter 1. The common implicit assumption is that collected sensor data is correct. However, given the real-world limitations we described earlier, this assumption is generally not true. Thus, the role of our approach is to semi-automatically and transparently process the orientation sensor data of the mobile videos and then provide more accurate data to existing GIS/social media applications. Our processing system works as a middleware layer (the dashed line box in Figure 5.1) between the raw sensor database uploaded by mobile devices and the processed sensor database used by upstream applications.

To filter out data noise, we design an effective correction algorithm based on geospatial matching and optical flow analysis consisting of three steps.

- Step 1: For a specific frame, we first gather extra information, *i.e.*, a landmark position in the visual domain, from the mobile client. We match this information against GIS data sources to infer the target landmark's geo-location with the highest probability. By knowing the most possible

geographic coordinates of the captured landmark and the camera, we estimate an accurate orientation.

- Step 2: Subsequently, we process two more steps to propagate the corrected orientation values from one specific frame to the whole video. By leveraging its consistency in the temporal domain, we compute the horizontal motion flows to interpolate highly accurate orientation data for every frame.

- Step 3: In order to decrease the accumulative errors during interpolation, we perform landmark matching between sampled frames to update the target landmark's visual position at a given rate.

5.2.1 Problem Formulation

In our context, orientation data of a mobile video is a time-series dataset consisting of compass reading values. Let $\Theta = \{\theta_1, \theta_2, \cdots, \theta_n\}$ and $\mathcal{F} = \{f_1, f_2, \cdots, f_n\}$ be the sequences of compass readings and their corresponding video frames for every time instance $\mathcal{T} = \{t_1, t_2, \cdots, t_n\}$, respectively. We denote the ground truth of the orientation sequence data as $\mathcal{G} = \{g_1, g_2, \cdots, g_n\}$. Both g_i and θ_i have values in the range 0 to 360 degrees. The direction measurement error for θ_i is the angle difference between its true and measured orientation $\delta_i = min(\|g_i - \theta_i\|, 360 - \|g_i - \theta_i\|)$. The direction error of Θ is the average of every sample's direction error, *i.e.*, $E_\Theta = \frac{1}{n}\sum_{i=1}^{n}\delta_i$. In the still image case, all data (orientation value Θ, frame content $\mathcal{F}$ and direction error E_Θ) only exist for one time instant $\mathcal{T} = \{t_1\}$.

Problem Statement: Given a sequence of orientation readings Θ and their related timestamps $\mathcal{T}$ and frames $\mathcal{F}$, find a sequence of estimated di-

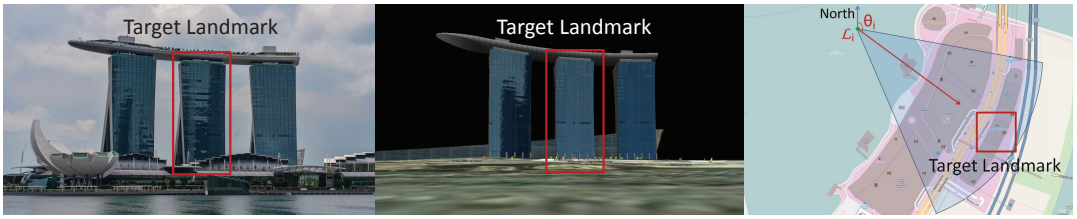


Figure 5.2: LEFT: A snapshot from one video that shows an outstanding architecture around Singapore Marina Bay; MIDDLE: the corresponding view of Google Earth 3D buildings from the same location and camera orientation; RIGHT: the FOV scene model of this frame illustrated on a 2D map synthesized by the same location and camera orientation.

rectional values, $F : \theta_i \rightarrow \tau_i$, such that the accuracy of processed orientation sequence $\Theta' = \{\tau_1, \tau_2, ..., \tau_n\}$ is enhanced by having $E_\Theta \not\leq E_{\Theta'}$, where $E_{\Theta'} = \frac{1}{n}\sum_{i=1}^{n} \delta'_i$ and δ'_i is each processed orientation's distance to the ground truth.

5.2.2 Geospatial Matching and Landmark Ranking

The orientation of a vector θ is determined by two distinct points in 2D geometry and we can obtain the position of the camera from the embedded GPS receiver. Therefore, our key idea is to estimate the real geo-location of a specific building appearing visually in a given still image or a video frame. As illustrated in Figure 5.2, by matching the building's position in the image with its position on a 2D map, the accurate orientation of the camera can be extracted through geometrical computations. Therefore, a landmark appeared in the picture or frame is required to develop our approach. Next we explain the detailed geospatial matching and landmark ranking procedure to determine the most possible geo-location of the target landmark in the scene.

Target Landmark Determination

It is fundamental to determine which landmark to match between the pixel and the geospatial domains. As a first step we customized the recording app *GeoVid* for both iOS and Android platforms. Our system includes a convenient interface to let users indicate the target landmark, which could be any kind of structure, with an easy touch-gesture input. For still image capture, we allow users to indicate the width boundaries of a close and prominent structure by moving their fingers along a landmark's vertical edges on the just-taken picture (see the yellow lines on the sides of the left tower in Figure 5.3). For video, users can also indicate a pair of perpendicular lines with the same touch-and-move operation at the beginning of a recording. Since we are concerned with the horizontal direction and angle change, the vertical edges provide the best information to track camera movements horizontally. Notably, through this user interface, we not only gather the target identification, but also quantify the marked buildings' visual width and degree of horizontal visibility with angle ranges, which will later be used for landmark ranking. Moreover, the image position of the user-indicated building is recorded as well to calculate the horizontal visual offset distance later.

Geospatial Matching

The next step is to determine all possible structures that may have been captured in the video, and rank those buildings to locate the one indicated by a user in the first phase with the highest confidence. Then we can utilize two distinct points, *i.e.*, this building's geo-location and the mobile device's coordinates, to compute the camera orientation. In our system, we retrieve geographic in-

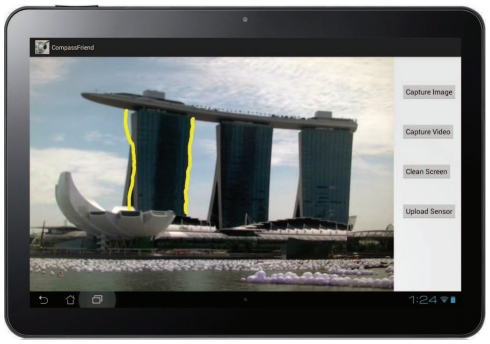


Figure 5.3: Image/video capture interface in modified *GeoVid* apps on iOS and Android platforms. The yellow lines are achieved through users' gesture-input, *i.e.*, finger swiping.

formation about surrounding structures from *OpenStreetMap* which is an open and free service built with voluntary contributions from users. This gazetteer provides comprehensive information, *e.g.*, name, type, location and polygon outline of each building in geospace. In order to find all buildings that likely appear in the scene, we query the geo-information service and retrieve all the architectures within a bounded distance $\mathcal{R}$ around the camera location point $\mathcal{L}$.

However, not all buildings that are retrieved are visible from $\mathcal{L}$, due to horizontal and vertical occlusions. To filter out these fully or partially occluded objects is a challenge. Inspired by the studies of Shen et al. [104] and Lee et al. [67], we designed a 3D visibility filter to exclude obstructed objects and obtain only the visible building candidates before the ranking step. Lee's method only considers the visibility occlusion relations within 2D space, based only on the outline of the footprints of the objects. Thus in their algorithm, vertically extensive (*e.g.*, tall and prominent) buildings would be considered as occluded by front objects and filtered out as noise. In Shen's study, they utilized the 3D context but did not consider the effects of horizontal occlusions. In our situa-

tion, users generally only mark a target building which is totally unblocked or at least partially vertically visible, because they need to indicate the vertical boundaries of the building. If an architecture is partially horizontally occluded, either one or even both of the vertical edges are not drawable. Hence it is necessary to improve Shen's method to satisfy the requirements of our system.

In our filtering method, we first classify all retrieved buildings into four categories according to their visibility levels:

- Category 1 – *Unoccluded Building*: A building to which we can cast a line-of-sight horizontal tracing ray from the camera location which does not intersect with any other objects and is completely visible by the lens.

- Category 2 – *Partially Vertically Visible Building*: A building which is partially occluded but is vertically visible since its height exceeds any front object blockage. When the vertical viewable angle of the object is greater than the angle of the object right in front of it, the object will be marked as partially vertically visible. Note that a building can be occluded by several other objects but still by partially vertically visible.

- Category 3 – *Partially Horizontally Visible Building*: A building which is partially occluded but is horizontally visible since its width exceeds what the front object is blocking. When the horizontal viewable angle of the object is greater than the angle of the object in front of it, the object will be marked as partially horizontally visible. Again, a building can be occluded by several other objects but still partially horizontally visible.

- Category 4 – *Occluded Building*: The building is neither unoccluded nor partially visible, therefore it is considered completely occluded.

Input: The camera location $\mathcal{L}$ of a FOV scene, and a finite set $\mathcal{B} = \{\omega_1, \omega_2, \ldots, \omega_n\}$ of retrieved buildings within distance $\mathcal{R}$ from $\mathcal{L}$
Output: A set of visible buildings and their horizontal visible angle range $\mathcal{O} = \{(\omega : [\mu, \nu])\}$

$\mathcal{O} \leftarrow \{\emptyset\}$
foreach $\omega_i \in \mathcal{B}$ **do**
  if $\mathcal{L} \in \omega_i$ **then**
    $\mathcal{B} \leftarrow \mathcal{B} - \omega_i$
  end if
end for
foreach $\omega_i \in \mathcal{B}$ **do**
  $(\mu_i, \nu_i) \leftarrow$ HorizontalRange$(\mathcal{L}, \omega)$
  $(\hat{\mu}_i, \hat{\nu}_i) \leftarrow$ VerticalRange$(\mathcal{L}, \omega_i)$
  $OccluFlag \leftarrow$ false
  foreach $\omega_j \in \mathcal{B}$ & distance$(\mathcal{L}, \omega_j) <$ distance$(\mathcal{L}, \omega_i)$ **do**
    $(\mu_j, \nu_j) \leftarrow$ HorizontalRange$(\mathcal{L}, \omega_j)$
    if $(\mu_j, \nu_j) \cap (\mu_i, \nu_i) = \emptyset$ **then**
      go to next loop
    end if
    $(\hat{\mu}_j, \hat{\nu}_j) \leftarrow$ VerticalRange$(\mathcal{L}, \omega_j)$
    if $(\hat{\mu}_i, \hat{\nu}_i) \subset (\hat{\mu}_j, \hat{\nu}_j)$ **then**
      $OccluFlag \leftarrow$ true
      leave loop
    end if
  end for
  if OccluFlag = false **then**
    $\mathcal{O} \leftarrow \{\omega_i : [\mu_i, \nu_i]\}$
  end if
end for
return $\mathcal{O}$

Algorithm 2: 3D Visibility Filter

Given a set of retrieved buildings $\mathcal{B}$, we filter out the third and fourth categories as noise since these two types of buildings would not be marked as target landmarks. The output of our filtering method is the set of visible or partially vertically visible objects and their horizontal visible angle range, *i.e.*, $\mathcal{O} = \{(\omega : [\mu, \nu])\}$. Algorithm 2 sketches the overall procedure of our 3D visibility filtering method. First, the buildings whose 2D footprint-polygons contain the location of the camera are filtered from further computation to

improve the performance. Because these buildings that encapsulate the camera location block almost 360 degrees of the horizontal angle range from a camera's FOV scene in 2D, while in the real case users are likely able to see the target landmark which they indicated previously. Next, for each landmark candidate, we examine whether they are horizontally occluded by the buildings which are closer to the camera location. The occlusions are calculated by intersecting the viewable ranges. If they are horizontally occluded, either partially or totally, we further examine the vertical occlusiveness between each pair of buildings with the occlusive relation. All structures determined as vertically invisible by horizontally-occlusive front buildings in this step will be excluded. Thus, only the buildings belonging to the first or second visibility category will be extracted into the output set of our filtering method.

Landmark Ranking

The above filtering step produces a set of landmark candidates among which we need to select the target landmark indicated by users. Given a building set $\mathcal{O}$ and a camera's raw sensor data triplets $\langle \mathcal{L}, \theta, \alpha \rangle$, we devise a ranking algorithm that computes for every building in $\mathcal{O}$ the probability of it being the target landmark. First we assess and quantify the relevance of a visible structure in each individual FOV scene according to the three relevance criteria below.

- *Closeness to the camera location*: A closer object is likely to be more conspicuous in a FOV scene. We formulate the score for the distance criterion as a Gaussian function [124]

$$pd_{\omega} = \frac{1}{\sigma_d \sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{distance(center(\omega),\mathcal{L})}{\sigma_d}\right)^2},$$

where $center()$ returns the coordinates of the building center, and $distance()$ computes the distance from $\mathcal{L}$ to the building center.

- *Closeness to the initial FOV scene center*: The raw sensor data is utilized to increase the confidence of building candidates within a certain rotation range. Additionally, according to an observation that people tend to focus on the center of an image [52], we promote the candidates whose horizontal visible angle range is closer to the raw camera orientation, which is the center of the FOV scene, and score it with formula [84]

$$pc_{\omega} = \frac{1}{\sigma_c\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{\|middle([\mu,\nu])-\theta\|}{\sigma_c}\right)^2},$$

where $middle()$ returns the middle angle of the horizontal visible range $[\mu, \nu]$.

- *Closeness to the real viewable range of the indicated building*: Since the user has indicated the horizontal edges of the target building in the pixel domain, we are able to infer the viewable range of the target building in 2D geometry space. We compare this information to the horizontal visible angle range of all candidates. The more similar the visible range between the target and the candidate, the higher the probability of the candidate being the target.

$$pr_{\omega} = \frac{1}{\sigma_r\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{\|\|\mu-\nu\|-RG_{target}\|}{\sigma_r}\right)^2}$$

$$RG_{target} = \frac{W_{target}}{R_h}\alpha,$$

where W_{target} is the width of the target landmark measured in pixels, R_h

is the horizontal length in pixels of the image or video frame[1], and α is the viewable angle.

With the assumption that the above three criteria are independent, we multiplicatively combine the obtained scores. Thus, the overall probability of a building candidate ω is a three dimensional Gaussian function.

$$p_\omega : \mathbb{R} \to (0,1),\ p_\omega = pd_\omega \cdot pc_\omega \cdot pr_\omega$$

$$\begin{aligned} p_\omega = A \cdot exp\Bigg(& -\frac{distance(center(\omega), \mathcal{L})^2}{2{\sigma_d}^2} \\ & -\frac{\|middle([\mu,\nu]) - \theta\|^2}{2{\sigma_c}^2} \\ & -\frac{\big\|\|\mu-\nu\| - RG_{target}\big\|^2}{2{\sigma_r}^2}\Bigg) \end{aligned} \tag{5.1}$$

Coefficient A is determined from the previous three criteria and the three σ values have physical meanings. They indicate the amount of noise power in the difference between the candidate measurements and the input values, and also represent how sharp the probability curve decreases when the difference becomes larger. In our implementation, we set $\sigma_d = 2,000$, $\sigma_c = 180$ and $\sigma_r = 1$ empirically.

Finally we select and present the top K building candidates with the highest probabilities after the ranking step and let the user choose the true corresponding *target landmark* among these K options. This simple user feedback scheme improves the system's target landmark determination accuracy. In our

[1] R_h is the resolution value on either the $x-$ or the $y-$axis, depending on the recoding pose which could be Portrait or Landscape mode.

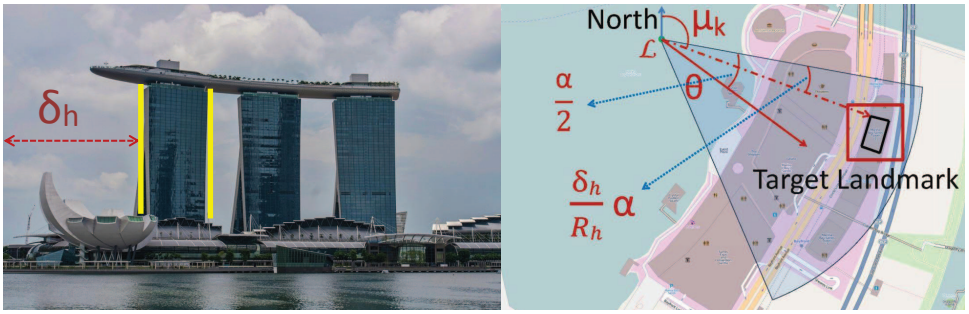


Figure 5.4: Orientation estimation based on target landmark matching between the geospatial and visual domains.

experiments, we display the top 10 candidates to users, and in our experience the target landmarks generally appear in the list.

The FOV scene information is calculated based on the sensor data which identifies the geographic region covered in the image. Thus, vice versa, with the known building reference in both the geospatial and pixel domains, we can accurately estimate the camera orientation on a 2D map with the following equation

$$\tau = \mu_k + \frac{\alpha}{2} - \frac{\delta_h}{R_h}\alpha \tag{5.2}$$

where δ_h is the target landmark's horizontal offset to the left boundary in the image and μ_k is the viewable angle range's left boundary of the target landmark (see Figure 5.4). In Equation 5.2, $\frac{\alpha}{R_h}$ indicates how many degrees the camera needs to rotate to make one pixel move in the image, and $\frac{\delta_h}{R_h}\alpha$ presents the degrees which the camera has to rotate to let the left edge of the target building to move from the image left boundary to the current position.

5.2.3 Landmark Tracking

After performing the above geospatial matching and landmark ranking algorithms, we are able to output one estimated directional value, $\theta_1 \rightarrow \tau_1$, for a still image or a specific video frame at time $\mathcal{T} = \{t_1\}$. During video recording, in order to minimize a user's required interaction, our system continues to track the interesting feature points detected around the target landmark to continuously calculate the position of this building in the next several seconds of frames. Afterward we use an affine model to estimate the target landmark's 2D transformation in the image and extract motion vector information on the horizontal axis, termed $x_i \cdots x_j$, to compute camera orientation values $\tau_i \cdots \tau_j$ for this portion of frames $f_i \cdots f_j$ (see Equation 5.3). Since we perform one visual feature tracking procedure between every two GPS signal updates, it is reasonable to assume that the camera location does not move too much within the tracking (*GeoVid* app updates GPS location only when the camera moves above 10 meters away from the previous record), and the camera is approximately performing a panning operation during the tracking period. Thus, we can estimate the orientation values by the equation below,

$$\tau_i = \frac{-x_i}{R_h}\alpha + \tau_{i-t} \tag{5.3}$$

Similar to Equation 5.2, we calculate the relative camera rotation first and add it to the previous estimated directional value to obtain the current frame's camera orientation. In motion vector notation convention, if the reference object moves towards the right in the image, the motion vector should be positive. While in the orientation notation, this case indicates that the camera is panning left (rotating counter-clockwise from an aerial view), which generates a negative

value accordingly. That is the reason why we change the sign to (-1) for the horizontal motion vector. Due to performance concerns, we do not compute the motion vectors between every two consecutive frames. Instead, we perform such tracking computation every t frames.

We use the Kanade-Lucas-Tomasi feature tracker [105] to infer motion vector x_i between f_i and f_{i-t}. Our system chooses and locates features by examining the minimum eigenvalue of each 2×2 gradient matrix, and features are tracked using a Newton-Raphson method of minimizing the difference between the two windows. An affine transformation is fit between the image of the currently tracked feature f_i and its image from a non-consecutive previous frame f_{i-t}. If the affine compensated image is too dissimilar, the previous extracted features will be dropped and new qualified features will be selected based on the same algorithm for substitution. Therefore in each motion vector calculation, we maintain a consistent number of tracking feature points FN through abandonment and replacement operations. Our implementation uses $t = 15$ and $FN = 150$ by considering both image size and performance.

When one tracking step is finished, our system performs orientation recalculation (Equation 5.2) again based on an updated camera location and an updated horizontal offset. We use the updated μ_k value by feeding the newest camera location into the $HorizontalRange()$ function (see Algorithm 2). The horizontal offset value is refreshed by accumulating all horizontal motion vectors x obtained in this procedure to the previous δ_h. When we detect that the target landmark is moving out of the viewable scene, *i.e.*, $\delta_h \geq R_h - W_{target}$ or $\delta_h \leq 0$, our system changes to track the feature points detected from the whole frame and extracts motion vectors based on these extended features. Lastly, all camera orientation values between τ_i and τ_{i-t} are estimated by linear

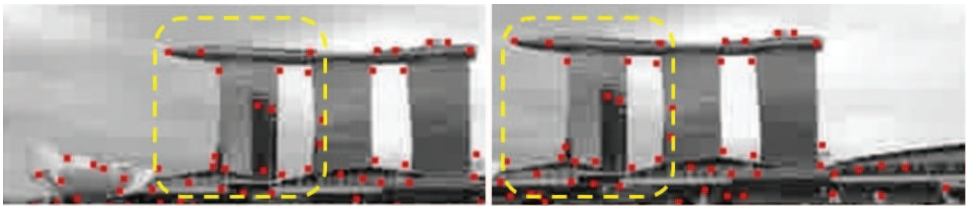


Figure 5.5: Illustration of landmark matching technique. Left: the original frame in which a user indicates a landmark. Right: a later, sampled frame in which the previously indicated landmark is expected to be recognized and localized.

interpolation. Our system continues such operations until the end of the video sensor file.

5.2.4 Sampled Frame Matching

Since the camera scenes are changing as time progresses and the estimated orientation output based on the tracking algorithm is precise only for scenes with the target landmark inside the image, our system needs to update the target landmark's visual position at a certain frequency in case the target moves in and out of the viewable scene. Additionally, errors are inevitably accumulated during the landmark tracking process. Such errors could be reduced by letting a user indicate the landmark multiple times as the video is recorded, but it would be too cumbersome for users. In order to maintain the correction accuracy over time and not burden users, we apply an object recognition technique to locate the target landmark in the sampled frames. If the target landmark is successfully recognized and localized in a frame, then it can be considered equivalent to a user input that updates a landmark's visual position. Hence, this terminates the previous and restarts a new landmark tracking process based on the re-estimated camera orientation.

We perform object matching through feature detection, extraction, and

matching followed by an estimation of the geometric transformation using the RANdom SAmple Consensus (RANSAC) algorithm. As illustrated in Figure 5.5, on the left is the frame in which a user indicated a landmark, and on the right is a sampled frame in which the landmark was automatically located since it appears. To match the landmark, we extract keypoints from both frames using SIFT descriptors [81]. A SIFT descriptor is a 128 dimensional feature vector that encodes the image information in a localized set of gradient orientation histograms. Sampling is performed in a regular grid of 16×16 locations covering the interest region. For each sampled location, the gradient orientation is entered into a coarser 4×4 grid of gradient orientation histograms with 8 orientation bins.

After the SIFT feature extraction, we compute the bounding box of the user indicated landmark in the left frame by adding a margin to the boundary drawn by the user. Next, we find the best candidate match for each keypoint within the bounding box by identifying its nearest neighbor among the keypoints from the right frame. However, local descriptor matching can produce many false matches. In order to reject such false matches, we first compute the ratio of the closest to the second-closest neighbors of each keypoint and only accept matches in which the distance ratio is less than 0.8, as proposed by Lowe [81]. Next we estimate the geometric transformation using the RANSAC algorithm, which can robustly fit a model to data in the presence of outliers and has been used to find correspondences in the presence of noise [32]. Iteratively, we randomly select a subset of keypoint matches, based on which we compute the transformation matrix for affine homography. The affine homography model is then tested against all the other keypoint matches. Matches that fit the model are considered as hypothetical inliers while the others are

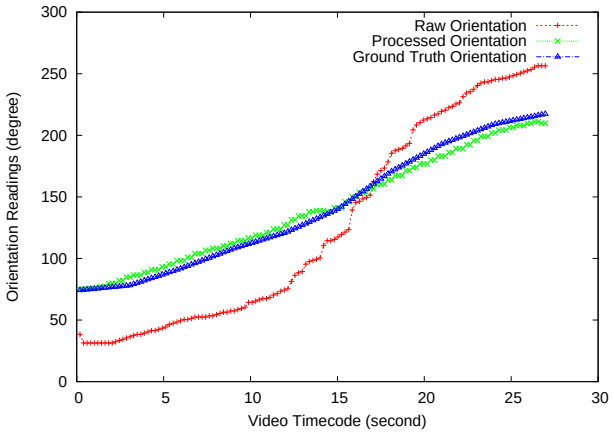
considered as hypothetical outliers. After a default number of iterations, the model with highest number of hypothetical inliers is selected.

For the selected model, we use a threshold $Th = 0.15$ to determine whether the target landmark is recognized in the right frame. If the number of inliers is smaller than Th, the landmark is considered to be absent. Otherwise, the inliers can be regarded as true matches between local features and the landmark is considered to be present. Next we translate the upper-left and lower-right points of the bounding box in the left frame using the same affine model, and regard them as the estimated boundary of the new bounding box for the recognized landmark in the right frame. Finally, we obtain the updated landmark position, and are able to re-estimate the camera orientation for this sampled frame.

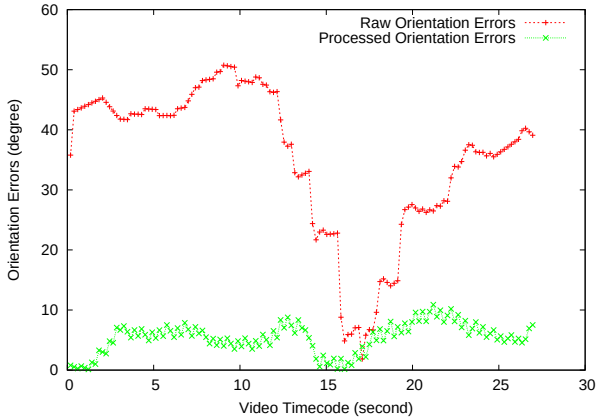
5.3 Experiments

In our experiments, we utilize the publicly available real-world georeferenced video dataset from the GeoVid website. We process the corresponding sensor data of the videos with our proposed methods and compare the results with Structure from Motion (SfM) and the ground truth, in terms of performance and accuracy enhancement, respectively. We randomly select 15 georeferenced videos recorded in Singapore where the sensor data is recorded by up-to-date mobile hardware (see dataset description in Table 5.1).

To obtain the ground truth data we provide two alternative ways for users to manually annotate true camera orientation values. For a given video frame, we first provide multiple Google Street View images and a Google Earth 3D synthesized view from the current GPS location. Users can compare the vi-



(a)

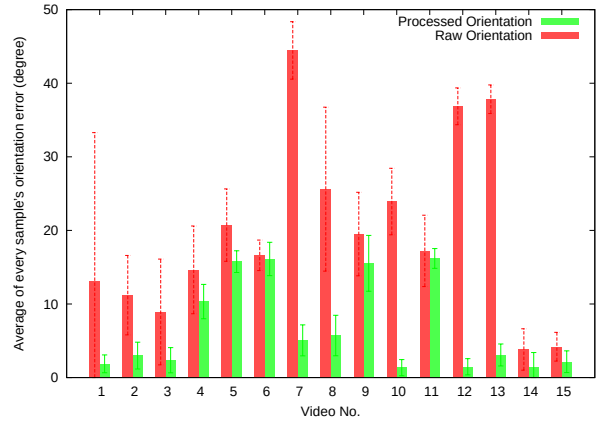


(b)

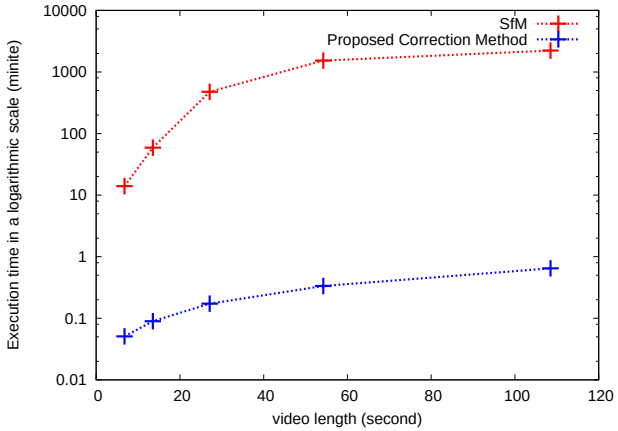
Figure 5.6: (a) Raw, processed and ground truth camera orientation reading results of one sensor data file (θ_i, τ_i, and g_i). (b) Raw and processed camera orientation error of each sample in one sensor data file (δ_i and δ'_i).

sual contents between the frame and the referenced views to determine the orientation value. In addition, we also allow users to indicate the geographic object that appears in the frame center on the Google Earth interface. The coordinates of the indicated object as well as the camera location is later entered into the Geotools library[2] to calculate the true camera orientation. For each experimental video, we sample frames every 3 seconds for users to perform the ground truth annotation. We interpolate the orientation degrees between

[2] http://geotools-php.org/



(a)



(b)

Figure 5.7: (a) Raw and processed camera orientation average error results of each video's sensor data (E_Θ and E'_Θ). (b) Execution time comparison between our proposed orientation data correction method and the SfM technique on a logarithmic scale.

sampled frames for later comparisons.

5.3.1 Accuracy Enhancement

We treat the first frame of each video clip as a still image and perform geospatial matching and landmark ranking algorithms on it. To the following frames, we apply the landmark tracking and sample frame matching methods. As will be shown in our experimental results, the system works well for mobile media.

We employ the *FFmpeg* library[3] to extract frames from the video dataset at a chosen resolution of 360 × 240 per frame to reduce the time consumption for image processing.

First we present the results for one camera orientation sequence. We apply our approach to one video's corresponding sensor data file. Figure 5.6(a) illustrates θ_i, τ_i, and g_i of the mobile test video and Figure 5.6(b) illustrates the errors δ_i and δ'_i accordingly. As shown, the raw orientation readings along the whole file are very much incorrect, which is causing the drift phenomenon when displayed on a map interface. After the correction by our algorithm, we find a distinct improvement such that the processed orientation data approaches the ground truth values and the error of each sample is considerably reduced.

Next, to quantitatively evaluate the camera orientation accuracy enhancement with our proposed method, we processed the sensor data of 15 videos (each sensor data record contains a sequence of FOV scene triplets) and compared the average direction error between the processed and raw orientation readings (see Figure 5.7(a)). On average, E_Θ of the raw orientation readings is 19.9 degrees and E'_Θ of the processed data is 6.76 degrees. Thus, our system significantly increases the accuracy of the camera orientation data by up to 66%. For some videos, *e.g.*, No. 7, 12 and 13, our correction algorithm enhances the accuracy from an approximately 40-degrees error to a level less than 10 degrees. However, in some other cases, such as No. 6 and 11, our method does not improve the accuracy much. We also report the feedback ranking results of the user indicated target landmarks in Table 5.2. No target is reported missing, which means users can always find their previously visually marked geo-object among the top 10 candidates returned by our ranking algorithm.

[3] http://ffmpeg.org/

Table 5.1: Georeferenced video dataset description.

Shortest video length	Longest video length	Average video length
11 sec	1 min 50 sec	23 sec
Light conditions	Phone models	
2 in evening, 13 during day	iPhone 4S and HTC Desire S	

The low average rank also indicates the high precision of our method for target landmark determination in geospace.

5.3.2 Performance

To the best of our knowledge there exists no other system that estimates the camera direction for the explicit purpose of camera orientation correction. We found the SfM technique to be the closest, related method that can output estimated camera poses as an auxiliary effect during the 3D structure reconstruction from a set of images around a landmark. Hence, we measure and compare the execution time between our method and an existing SfM system [109] to evaluate the efficiency of our proposed approach. We process georeferenced videos of different lengths with both systems. For our proposed orientation data correction method, we perform all measurements on a 3.4 GHz Intel Core i7-2600 CPU with 4 cores and 8 GB of memory.

We apply SfM on our dataset as follows. First, from the frame dataset we extract features with the SIFT method from the VLFeat library. Afterwards, feature matching and bundle adjustment are performed with the SfM bundler library [109]. Next the output of the SfM step is fed into CMVS (Clustering Views for Multi-view Stereo) to divide the image set into clusters of manageable size and allow them to be processed independently and in parallel [33]. Eventually the PMVS2 (Patch-based Multi-view Stereo) software is executed

Table 5.2: Target landmark ranking results from users' feedback among 15 test videos.

Highest rank	Lowest rank	Average rank
1 (for 8 videos)	10 (for 1 video)	2.6

to produce a set of oriented points instead of a polygonal (or a mesh) model, where both the 3D coordinates and the surface normals are estimated at each oriented point [34].

Since the SfM technique requires much more computing resources (known from reported experiments in other studies), we measure its performance with the same dataset but on a 2.67 GHz Intel Xeon X5650 CPU with 12 cores and 64 GB of memory. Figure 5.7(b) illustrates the execution time to process videos of different input lengths. As shown, even though the SfM system utilizes more hardware resources, it takes orders of magnitude longer to compute the camera orientation compared to our method. For lengthy sensor data files, *e.g.*, videos longer than 100 seconds, the processing time of SfM is almost 3,000 times longer. Moreover, the camera orientation values from SfM (which can be obtained by an "up-right" vector estimation and a 2D-2D transformation [115]) in the experiments are even *less* accurate than the raw data, *i.e.*, $E_{\Theta'} > E_{\Theta}$. We believe the reason for this is that the frames from test dataset do not always provide a good cover of the reconstructed object. Moreover, the requirement of short-baseline stereo increases the difficulty of 3D model reconstruction and image registration for the SfM technique. The results show the capability of our system to perform correction effectively with a single video input in a reasonable amount of time.

Figure 5.8: Screenshot of the Oscor visualization interface.

5.4 Demo System

We implemented our framework and introduce a demo system named *Oscor*, which corrects orientation measurements for still images and video frames based on geographic analysis and image processing. With regards to video recording, in order to minimize the user interaction, we compute the horizontal motion flows and perform landmark matching between sampled frames to interpolate highly accurate orientation data for every frame.

Figure 5.1 illustrates *Oscor*'s overall architecture. For media capture, we provide a camera app which acquires location and orientation information concurrently from multiple sensors while taking images or recording videos [123]. Moreover, our app presents a convenient interface to gather extra information, *e.g.*, a landmark scene position and its name, without requiring cumbersome input from users. For still image capture, we allow users to indicate the width boundary of a close and conspicuous structure by moving their fingers along a building's vertical edges on the just-taken picture. For the video, users can also indicate a pair of perpendicular lines with the same touch-and-move operation during recording. After receiving this edge information on the server side, we convert the users' input of a landmark's position from the pixel index domain

into the geospatial domain, and subsequently perform GIS analysis.

We employ OpenStreetMap (OSM) as our GIS reference database and query related building locations and their 2D shape polygons within a certain distance range to the camera position [104]. From the 2D shape information, our system computes each building's width value in the image's x-dimension from the camera's point of view. Afterwards, we utilize the building width information extracted from pixel level, the building distance to the camera from the GIS analysis, and the initial orientation measurement collected from the mobile-embedded digital compass, to build a 3D Gaussian model calculating each building's probability of being the landmark in the user's still picture or video frame. Subsequently, Our system returns the top K landmark names back to the user for selection of the correct one. With the known building reference in both the geospatial and pixel domains, we can accurately estimate the camera pose on a 2D map, and hence output the corrected camera orientation data for the corresponding image/frame.

On the app we have added a transparent overlay on top of the camera interface and leverage multi-touch gestures to collect the building position information indicated by users. After a user uploads the raw sensor data via an HTTP link, the server efficiently stores and indexes this information into a NoSQL MongoDB database.

The *Oscor* user interface visualizes the static or moving field-of-views of images and videos, which allows users to experience fused video browsing based on geographic properties. On a Google Maps canvas multiple images/videos are presented as pins. When a user clicks or touches a pin, a map-overlay image viewer/video player is launched and the video is rendered from the designated starting location. During video playback, the camera's current location and

viewable scenes are animated along the corresponding GPS trajectory. To help users visualize the corrected contextual information and how the erroneous data possibly effects further processes, two viewable scenes based on asynchronously retrieved raw and corrected camera orientation data are rendered on the same interface (see Figure 5.8).

5.5 Summary

We presented an approach for camera orientation data correction based on geospatial analysis and image processing techniques. We analyzed the viewable scenes of mobile videos and devised algorithms to estimate more precise orientation data. The experimental results demonstrate that our technique is very effective (improve the accuracy by up to 66%) and efficient (up to 3000 times faster) compared with the ground truth and an existing system. One limitation of our work is that the mobile media content must contain at least one geo-object, such as a landmark, for our system to perform matching in geospace, analysis and data correction. As part of our future work we plan to investigate other visual features and sensors embedded in mobile platforms to help with camera orientation correction without the restriction of a geo-object's presence.

CHAPTER 6

Sensor-assisted Camera Motion Characterization and Video Encoding

6.1 Introduction

In existing multimedia applications listed in Chapter 1, higher level semantic results can be computed from the very low level contextual information (i.e., sensor data). In this thesis, after we obtain more accurate and reliable sensor information, we explore the possibility of applying sensor analysis techniques to new mobile media applications, such as video encoding and 3D model reconstruction (see Figure 6.1). This Chapter describes how we leverage location and orientation sensor data analysis in video encoding improvement based on the camera motion characterization.

Camera motion is a distinct feature that essentially characterizes video

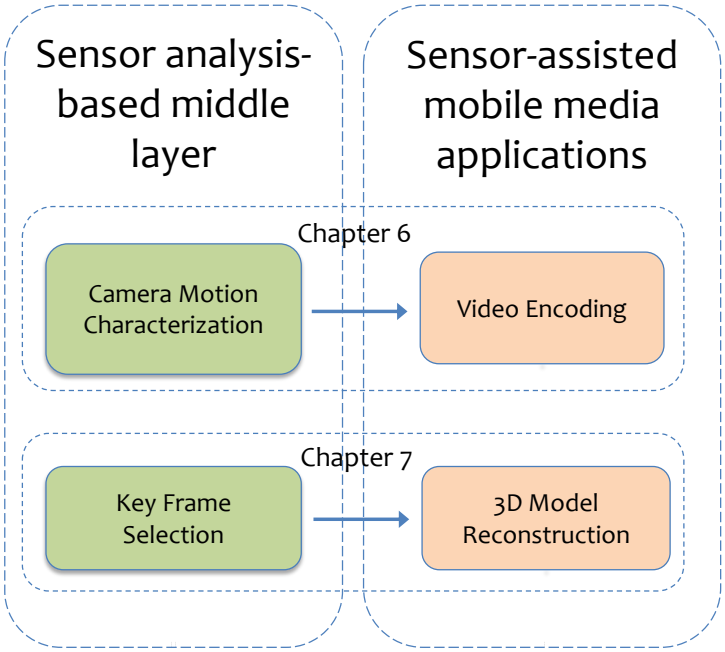


Figure 6.1: The proposed sensor-assisted applications.

content in the context of content-based video analysis. Almost all existing work relies on a content-based approach at the frame-signal level, which results in high complexity and very time-consuming processing. Similarly, capturing videos on mobile devices is also a compute-intensive and power-hungry process. One of the key compute-intensive modules in a video encoder is the motion estimation (ME). In modern video coding standards such as H.264/AVC, ME predicts the contents of a frame by matching blocks from multiple references and by exploring multiple block sizes. Not surprisingly, the computation and power cost of video encoding pose a significant challenge for video recording on mobile devices such as smartphones.

Our solution for addressing these two challenges is to perform sensor-assisted camera motion analysis and introduce a simplified motion estimation algorithm for H.264/AVC. We employ relatively low-power sensors to classify

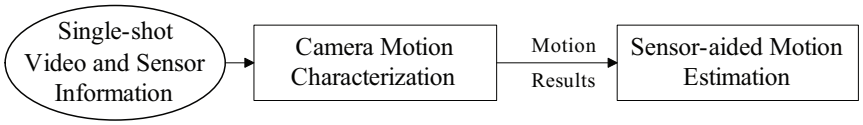


Figure 6.2: Overview of the proposed two-step framework.

camera motion types and subsequently apply the motion type information to significantly simplifying motion estimation. Our approach is motivated by two observations. First, modern smartphones include relatively low-power and low-cost sensors, such as GPS and digital compass. The geographical properties are provided by these sensors and their accuracy is enhanced by our post-processing methods discussed in Chapter 4 and 5. The corrected geographical properties are generally quite intrinsic to device motion characterization. Second, in many video documents, particularly in those captured by amateurs, a global motion is commonly involved owing to camera movement and shooting direction changes. In outdoor videos, *e.g.*, videos capturing landmarks or attractions, global motion contributes significantly to the motion of objects across frames.

Our method introduces a two-step process which is outlined in Figure 6.2. First, as a key feature we only use geographic information to detect subshot boundaries (within one shot, subshot is defined as a smaller unit, whose contents only contain one motion type) and to infer each subshot's camera motion type from the collected sensor data without any video content processing. With generated camera motion information, we modify the HEX motion estimation algorithm used in H.264 to reduce the search window size and block comparison time for different motion categories, respectively. Our experimental evaluations show that our motion characterization method can accurately segment subshots and label their classification, and our simplified motion estimation algorithm

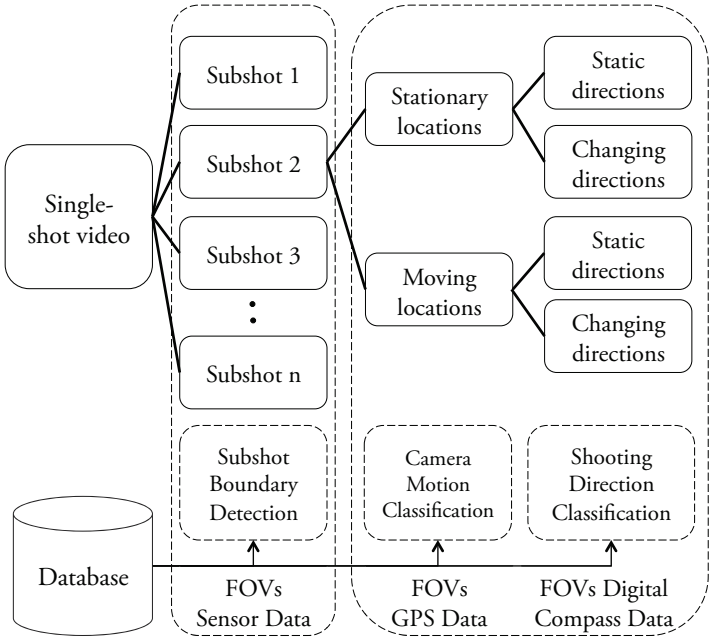


Figure 6.3: Proposed camera motion characterization framework.

can reduce the complexity of the H.264/AVC motion estimation with the HEX algorithm by up to 50% while speeding up the estimation component considerably.

6.2 Camera Motion Characterization

We utilize a stream of sensor data which is simultaneously collected with video frames to describe the geographic properties related to the camera view. In Chapter 3 we describe the viewable scene model used and present how we collected videos and their sensor measurements. This section describes our sensor data based approach to subshot boundary detection and camera motion characterization. Our simplified motion estimation algorithm that works by reducing the number of candidate blocks is presented in Section 6.3.

Our camera motion characterization system works in two steps as illustrated in Figure 6.3. First, we analyze all the sensor data of the acquired single-shot video to detect subshot boundaries. For each subshot segment we examine the camera location movement states and the shooting direction change states. Then we determine the relationship between the camera trajectories and way the the shooting direction changes. Using the interrelation analysis results, we are able to characterize the camera motion type for every subshot.

6.2.1 Subshot Boundary Detection

A camera's mobility is characterized as either *moving* or *stationary*. Similarly, during a given period of time, the shooting direction of the camera can also be either *fixed* or *in motion*. A specific camera motion type is a combination of two state components. For example, *Panning* consists of a stationary camera location and a particular change model the of shooting direction.

We first search subshot boundaries based on the camera location mobility. We calculate the GPS speed value of each FOV sample and binarize them as moving or stationary. To evaluate the direction state, we read the compass values and compute their smoothed directions as defined in Equation 6.1 to further seek for the subshot boundaries. The smoothed direction at time index t is a weighted average of the previous w and the next w direction values. If the processed direction exceeds a certain threshold T_d, we consider them as in motion. Next we select the start points of each group of consecutive frames who share the same location movement and direction change status as the boundaries of subshots. Using two different types of boundaries (the boundary between moving and stationary camera location, and the boundary between

$\mathcal{L}$ \ α	Quasi-static	Changing
Stationary	*Still*	*Panning, Tilting*
Moving	*Tracking, Dolly in/out*	*Focusing, Scanning*

Table 6.1: Semantic classification of camera motion patterns based on a stream of location $\mathcal{L}$ and camera direction α data.

changing and static shooting direction), we divide the video into subshots and classify each of them by its camera motion category. We further set a threshold T_{temp} as the minimum segment-duration by observing that a camera motion is generally maintained for at least several seconds.

$$a_t = \frac{\sum_{i=-w}^{w} p_i \times a_{t+i}}{2w+1} \tag{6.1}$$

Here $2w+1$ is the window size and p_i is the accuracy weight. If $a_t \geq T_d$, t is chosen as one of the subshot boundaries.

6.2.2 Subshot Motion Semantic Classification

After a coarse classification on both location movement and shooting direction change, we assign each subshot's camera motion type further to fine-grained classes. Specifically, we associate the relationship between the moving directions of the camera and its corresponding shooting directions. For each segment we obtained from previous step, we first compute the directions of camera movements. Along the trajectory, we employ the GPS values with a certain sampling rate, and achieve the camera moving direction of every segment by calculating the angle of vectors, which consists of a start location and an end location. Afterwards we are able to compare the relation between the moving directions of camera and their corresponding shooting directions.

If there exists no significant fluctuation in both locations and shooting directions, we categorize it as *Still*. If only the shooting direction changes, a detected subshot can be labelled as *Panning* or *Tilting*. With the help of the accelerometer sensor, we can easily detect a change of the lateral axis (pitch) and consider the shooting behavior as *Tilting*, otherwise, we mark it as *Panning*. If the location moves while the shooting direction is rather quasi-stationary (below a threshold) and the angle between several direction vectors is larger than L but less than $180 - L$ degrees (L is the angle degree border that separates different classifications) we label the shooting behavior at this point as *Tracking*. Otherwise, it will be considered a *Dolly in* (moving forward while shooting in the same direction) or *Dolly out* (moving backward while shooting in the opposite direction), respectively. When a camera's direction and location move simultaneously, it is difficult to clearly identify any useful patterns except possibly *Focusing* (pointing to a specific object). In such cases we term them as *Scanning* (our method does not distinguish those two at present time). In the scope of one segmentation, the majority of shooting behaviors are consolidated into the classification label of this sub-shot. In view of the source of sensor-tagged videos, which are mostly captured by smartphones, functions like zoom-in or zoom-out are currently not available during video recording on those devices.

After classification, each subshot belongs to one of the camera motion patterns (*Still*, *Panning*, *Tilting*, *Tracking*, *Dolly in/out*) with the categories listed in Table 6.1. Since we found the view direction values to be very noisy, we use their exponential moving average during the data analysis, which assigns higher weights to the latest measurement result.

6.3 Sensor-aided Motion Estimation

In video encoding phase, the camera motion information is mostly reflected in the global motion estimation (GME) model. MPEG-4 ASP supports GME with three reference points, although some implementations can only make use of one. GME can perform good estimation for global frame changes and supports different transformation types with very low complexity. However, it does not compensate for some local changes within a frame. Thus, some widely used video encoder implementations do not support GME well, *e.g.*, *x264*, an open source implementation for encoding video streams into the H.264/MPEG-4 AVC format. Instead, block-based motion estimation (BME) is extensively used in such software. Although BME is capable of achieving a good estimation of the local movement, it also incurs extremely high computational complexity. In order to shorten the processing time of BME, we apply the camera motion information generated in the previous step to simplify the HEX motion estimation, which is the default BME algorithm employed by *x264*.

In H.264, each macroblock is predicted from a block of equal size in the reference frame. The blocks are not transformed in any way apart from being shifted to the position of the predicted block. It is the motion estimation algorithms' responsibility to search and calculate this shift, which is represented by a motion vector. As illustrated in Figure 6.4, the HEX algorithm starts from the reference macroblock O predicted by the computed motion vector values of the left, top and top-right macroblocks of the prediction macroblock. Afterwards HEX iteratively compares the macroblocks around O with the prediction macroblock following the order from macroblock A to F, located in a hexagonal shape. In one iteration, HEX performs six macroblock comparisons

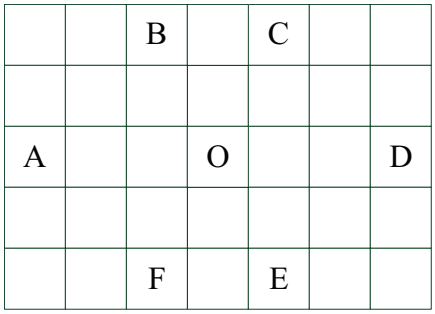


Figure 6.4: Illustration of the HEX Motion Estimation algorithm. Each grid represents a macroblock in the reference frame.

and considers the block with the minimum difference measurement result (*e.g.*, SAD) as the new search center in the next iteration. This search phase ends when the reference macroblock reaches the edge of the search window, or when it encounters a macroblock whose difference value is lower than a configured threshold.

Given the camera motion information computed earlier we reduce the search window size for each motion type. The reduction speeds up the search algorithm which is the most time-consuming part of the motion estimation.

- Class *Panning* and *Tracking*

  Since the shooting direction change and camera location movement both reflect horizontal translation of frames, the reference block stands a high chance of being located at the left or right of the prediction block. Thus, for these two classes, we narrow the search window into a flat rectangle with the same x-axis value as the prediction block, instead of a 16×16 square by default. As a result, in every difference calculation iteration, only block A and block D are compared with block O and used as reference block in the next iteration.

- Class *Tilting*

  In this case, no location movement is involved and the shooting directions include only tilts up and down, which results in a series of frames mostly containing vertical translations. Based on this information, we reduce the search window by ignoring the two blocks with the same x-axis index. Inside the narrow window we select blocks B, C, E and F as comparison blocks for the difference measurements in every iteration.

- Class *Dolly in/out*

  With these two classes, we are concerned about the macroblocks in the left half image and right half image separately. The rationale is that users tend to focus on objects located at the median plane of the FOV. Hence most of the blocks in the left part move towards straight left, top-left or bottom-left corner for a *Dolly in* camera motion. Therefore, when processing the macroblocks with an x-axis index less than half of the frame's width, we only estimate the left part of the search window, namely blocks A, B and F in each iteration. Similarly, prediction blocks located in the right half of the image would be compared with blocks C, D and E during every local search. A *Dolly out* segment looks like a reverse scan of the *Dolly in* motion. Accordingly, we switch the trimmed search window for the left half and right half macroblocks in the *Dolly in* case and apply them to the *Dolly out* pattern directly.

By simplifying the HEX algorithm with our method, most of the important motions of objects can be estimated much more efficiently. From our observation, in most of outdoor videos that capture landmarks or attractions, the local motion does not contribute a lot in the video content. Hence we

sacrifice a slight decrease in video quality to accelerate the motion estimation considerably. Since we apply different strategies to different motion patterns, our experimental results show that our compromise is reasonable and beneficial.

6.4 Experiments

6.4.1 Camera Motion Characterization

In our experiments, we apply our algorithms to sensor-annotated video dataset from the Geovid and we set parameters w=6 frames, and T_d=0.0054 degrees/sec empirically. We first compare the accuracy of our approach's camera motion classification to the ground-truth of the subshots, which was manually annotated. We report both the accuracy of the subshot boundary detection and the precision of the motion classification.

The first column in Table 6.2 shows the ground-truth time interval of each subshot and the second column illustrates the boundary times detected automatically by our system. The results match the ground-truth values very well. By comparing the start and end times of each classification's duration, we can see that the inaccuracy of our approach is generally ≤ 1 second. Note that some parts of the 1 second errors are contributed by rounding (because users generally cannot cut the video with an accuracy of less than 1 second, and the results of our approach need to be rounded off before the comparison). The results in the sixth and seventh row are the only boundary errors which are larger than 1 second compared to the ground-truth classification. The second-to-last row in the table represents an over-detection of subshots in our system. We observe that in the original video this part of the time interval does not

Ground-truth subshots	Subshots detected by our algorithm	Start time difference	End time difference
0:00 - 0:13	0:00 - 0:12	0	-1
0:13 - 0:17	0:12 - 0:17	-1	0
0:17 - 0:22	0:17 - 0:23	0	+1
0:22 - 0:24	0:23 - 0:24	+1	0
0:24 - 0:33	0:24 - 0:35	0	+2
0:33 - 0:51	0:35 - 0:50	+2	-1
0:51 - 0:57	0:50 - 0:57	-1	0
0:57 - 1:04	0:57 - 1:03	0	-1
1:04 - 1:07	1:03 - 1:06	-1	-1
1:07 - 1:12	1:06 - 1:12	-1	0
1:12 - 1:15	1:12 - 1:15	0	0
1:15 - 1:23	1:15 - 1:22	0	-1
1:23 - 1:30	1:22 - 1:29	-1	-1
1:30 - 1:38	1:29 - 1:38	-1	0
1:38 - 1:39	1:38 - 1:39	0	0
1:39 - 1:44	1:39 - 1:44	0	0
1:44 - 1:46	1:44 - 1:46	0	0
1:46 - 2:37	1:46 - 2:22	0	+1
	2:22 - 2:26		
	2:26 - 2:38		
2:37 - 2:41	2:38 - 2:41	+1	0

Table 6.2: Subshot classification comparison results of a sample video. The first column was obtained from manual observations, while the second column was computed by the proposed system.

appear as scanning behavior, rather a strong camera shake occuring when the operator climbs some stairs.

We apply our algorithm to nine sample videos randomly chosen from our video database. A summary of the quality of our classification method is presented in the format of a confusion matrix. As we can see in Table 6.3, the correctly classified outputs add up to 188 cases in total (the sum of the values across the diagonal) while incorrectly classified cases are 21. Therefore, the classification accuracy our approach can achieve is about 88%. Among the

G \ E	Still	Panning	Tracking	D/I	Scanning	D/O
Still	24	0	0	0	0	0
Panning	1	27	0	0	0	0
Tracking	2	0	16	1	2	0
D/I	0	0	1	49	6	1
Scanning	0	0	4	2	70	0
D/O	0	0	1	0	0	2

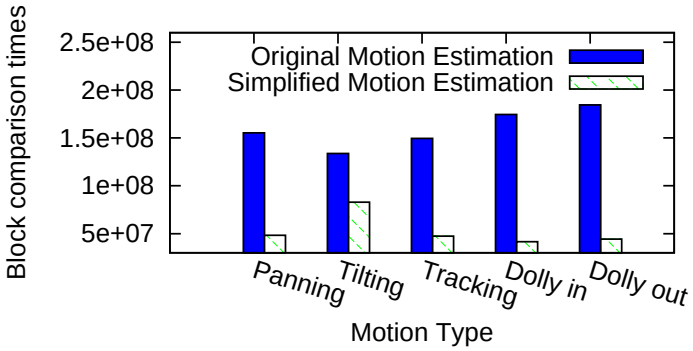
Table 6.3: Confusion matrix of our subshot classification method with nine sample videos. G represents the user-defined ground-truth, while E stands for the experimental result from our characterization algorithm. D/I and D/O are short for Dolly in and Dolly out respectively.

classified results, 40% are evaluated as scanning, which means that we cannot observe any meaningful semantics.

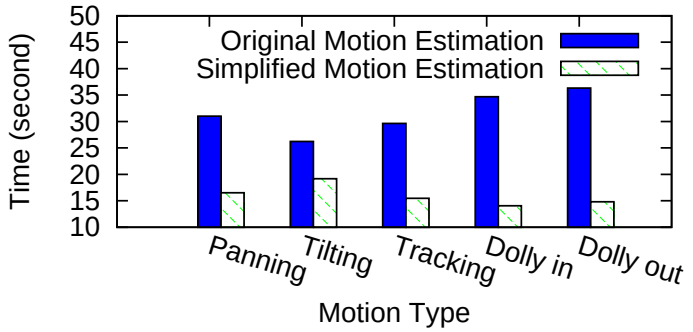
6.4.2 Sensor-aided Motion Estimation

The smartphones that we employed to record videos and sensor information in our experiments included models from Apple, HTC and Motorola. Videos captured by these mobile phones have a resolution of 720×480 or 1920×1080. The frame rate is either 30 fps or 24 fps. Since smartphones do not support raw video format recording, we converted the captured sequences into the YUV format with the *FFmpeg* tool. To implement our simplified motion estimation algorithm we modified the estimation functions in the source code of the *x264* codec software.

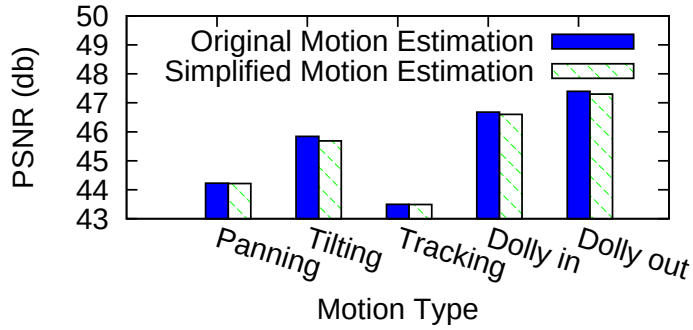
We report the macroblock comparison times in the estimation step and record the real time cost in the reference macroblock search and the block difference computation. Figures 6.5(a) and (b) show the results from our method and the original HEX algorithm. For every motion type processed differently by our approach, we successfully reduce the block comparison time and the real



(a) Block comparison time



(b) Motion estimation time



(c) PSNR

Figure 6.5: Macroblock comparison times, real time cost in the motion estimation algorithm and PSNR results for the original and simplified methods.

time cost in the motion estimation. Although we sacrifice some video quality, the results of the PSNR comparison (see Figure 6.5(c)) show that our method only introduces a relatively small decrease in quality. The reason of this very slight impact is that only minor local motion is involved in most of the outdoor

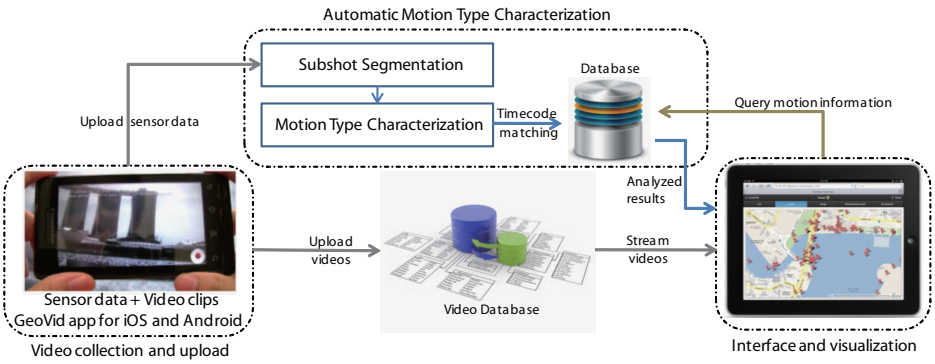


Figure 6.6: Architecture of the *Motch* system.

videos that capture landmarks and attractions.

6.5 Demo System for Camera Motion Characterization

Our demo system, *Motch*, efficiently segments videos and classifies each subshot's motion type purely based on the geographic sensor information of a video clip [125]. We utilize the open-source, NoSQL *MongoDB* database system [1] to store and index video motion types, and all motion information is wrapped as a RESTful service. When users upload sensor-rich videos to our server, the system immediately processes the associated sensor data and produces motion type results for these videos. Other applications are then able to utilize the motion information through our APIs and based on given video IDs. Finally, the *Motch* system provides an interactive interface compatible with browsers on both PCs and tablets. We dynamically present the visualization of spatial video scenes, motion type statistics and subshot details. Users are also able to scan through a video based on motion segmentation. Unlike traditional

[1] http://www.mongodb.org

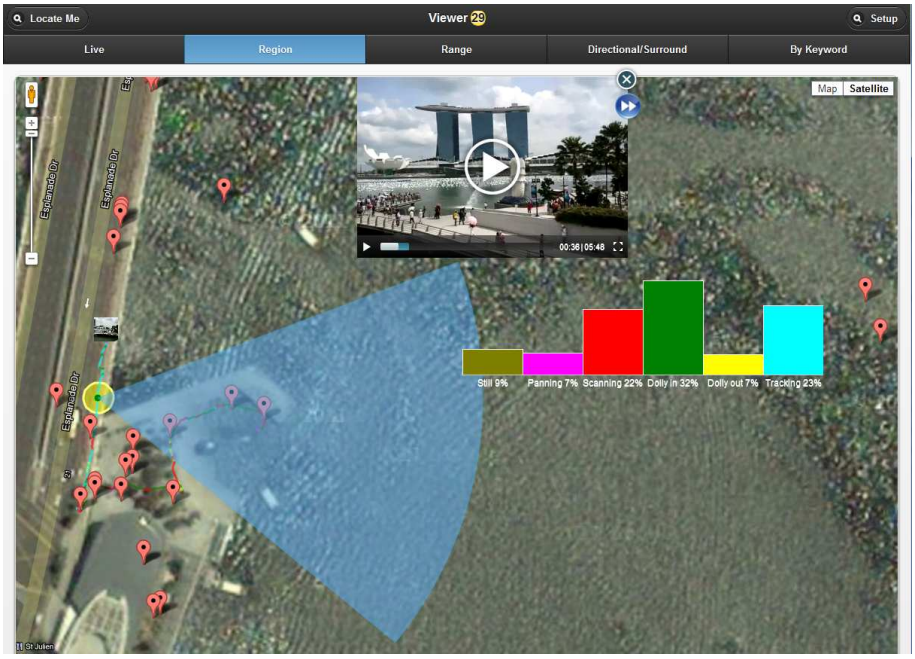


Figure 6.7: Screenshot of the *Motch* interface.

content-based methods, *Motch* does not analyze video content throughout the described process and is hence very efficient and fast.

Figure 6.6 illustrates *Motch*'s overall architecture. When users upload a video and its associated sensor measurements (in JSON format) to our *GeoVid* server from a mobile app, the server processes the GPS location and the viewing angle of every video frame and inserts them into a spatial database to make the information searchable. To enable effective wide-area network transmissions we transcode the videos to a lower 768 kbps bitrate and 480×360 pixel resolution.

The *Motch* user interface visualizes the processed motion type information for different video subshots and allows users to experience fused video browsing based on its geographic properties. Figure 6.7 illustrates the system interface. On a Google Maps canvas multiple videos are presented as pins, which indicate the beginning location of each video clip. The set of videos is automatically updated whenever a user navigates across the map. When the mouse pointer

hovers over one of the video pins, the corresponding GPS trajectory and motion type information are asynchronously retrieved from the server and displayed on the map interface.

When a user clicks or touches a pin, a map-overlaid video player is launched and the video is rendered from the designated starting location. During the playback of the video, the camera's current location and viewable scene are animated along the corresponding GPS trajectory, using the Raphaël vector graphics engine and HTML5 MediaElement APIs. Meanwhile, a motion classification statistics histogram of this video is presented below the player, and different subshots with various motion types are marked with different colors. All these results are computed and rendered in real time from our RESTful service output data. Moreover, users are able to click the "fast forward" button in the upper right corner of the player to directly jump to the next subshot with a different motion type. We also expose the motion type and related video time code information through web APIs to facilitate other video retrieval and analysis applications' access. Other researchers can also submit their own motion type characterization algorithms and hence visually evaluate their methods' quality through our system. Our interactive interface enhanced with visual features provides the user with a clear understanding of the various motion types in the video.

6.6 Summary

We propose a novel camera motion type characterization framework purely based on sensor data analysis, as well as a demo system, called *Motch*. Our method processes the sensor data collect by mobile devices to automatically

detect motion transition boundaries and to precisely classify the motion type of each video segment based on a camera movement and shooting direction change analysis. The system achieves highly accurate classification results from our experimental evaluation (around 88%). We also report on utilizing sensor information to simplify motion estimation in the H.264/AVC codec. With the proposed approach an almost equivalent PSNR performance can be maintained even with a much smaller search window for motion estimation. This leads to significantly reduced computations and therefore diminished hardware requirements and longer battery life for smartphones.

CHAPTER 7

Sensor-assisted Key Frame Selection for 3D Model Reconstruction

7.1 Introduction

Given the video contents that are automatically geo-tagged at the fine granularity of frames, another application that will benefit from our sensor data analysis is the automatic 3D reconstruction from the geo-tagged videos. This Chapter presents the methodology of the key frame selection from crowdsourced videos and the effectiveness and efficiency of our approach in later 3D model reconstruction phase.

Recently there has been significant progress in techniques that focus on recovering 3D scene geometry from multiple 2D images. Traditionally the images used for such purposes are carefully and specifically recorded to show the

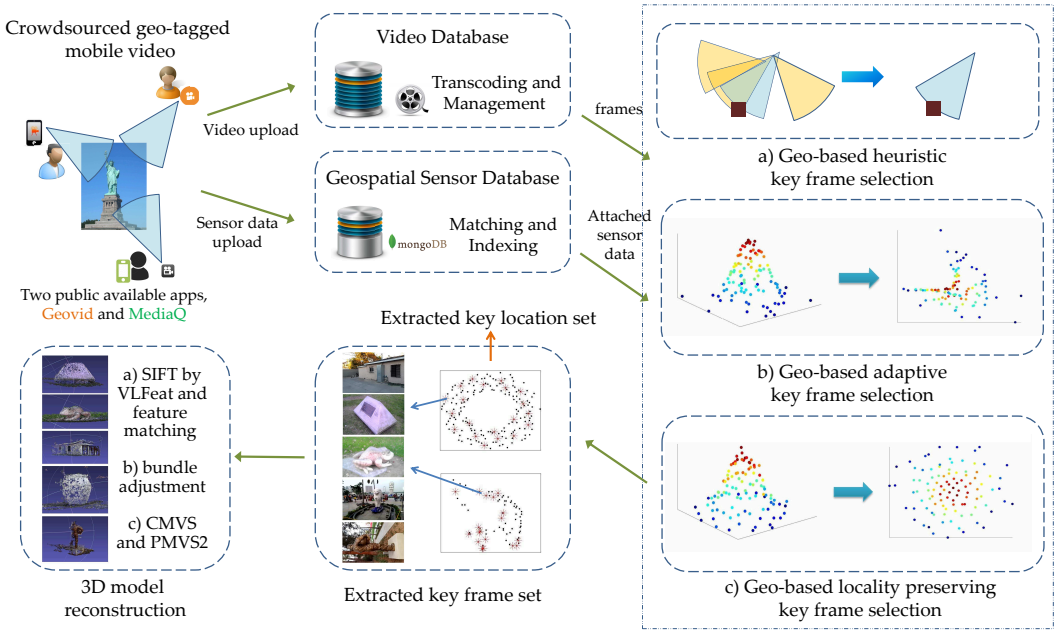


Figure 7.1: System overview and a pipeline of video/geospatial-sensor data processing.

candidate 3D object from different viewing angles while avoiding too much spatial overlap. It is important to obtain a near-optimal number of images from distinct viewing angles because too few images will result in visual "holes" in the reconstructed object, while too many images will unnecessarily increase the computational load and execution time and in some cases introduce artifacts.

Recently, many scenes (especially outdoors) are being captured from multiple viewpoints through UGVs. We explore the feasibility of using a set of UGVs to reconstruct 3D objects within an area. Such a method introduces the following challenges. First, videos are recorded at 25 or 30 frames per second and successive frames are very similar. Hence not all video frames should be used – rather, a set of *key frames* needs to be extracted that provide optimally sparse coverage of the candidate object. Second, the camera position and visual trajectory of UGVs are determined by the actions of an individual user. Such

videos are not usually captured with 3D reconstruction in mind.

To overcome above issues we leverage another technological trend – the geo-spatial metadata attached to videos at a fine-granular frame level. Figure 7.1 illustrates the system overview of the proposed 3D model reconstruction framework. We provide two mobile apps, *GeoVid* and *MediaQ* in the public market for users to record geo-tagged videos. These crowdsourced videos are afterwards uploaded to and transcoded in our cloud servers for 3D reconstruction use. All related geospatial sensor data are processed and indexed in our NoSQL database as well. For a given object, our key frame selection algorithms query the according georeferenced data, compute the most representative key frames and extract them from the video database. Finally, based on the informative key frame set efficiently determined by our method, we leverage an open-source structure-from-motion (SfM) library to reconstruct the 3D model of the target object.

The main component of this video and geospatial sensor data processing pipeline is the active key frame selection algorithm based on a manifold adaptive kernel and locality preserving reconstruction. To efficiently determine an effective set of key frames, we leverage (a) the available crowdsourced UGVs in the region and (b) the frame-attached geo-spatial metadata. In effect, our approach enables the *repurposing* of UGVs for 3D object reconstruction. Our algorithms select the most representative video frames with respect to the intrinsic geometrical structure of their geospatial data. We assume that each UGV frame and its geospatial neighbors lie close to a locally linear patch of the manifold. The manifold structure is characterized by the linear coefficients that reconstruct each video frame's geo-location from its neighbors. A transductive learning algorithm is applied to reconstruct the whole UGV set. The

most representative UGV frames are selected whose geo-location is chosen to reconstruct the original frame set best. Our experimental results demonstrate not only the computational feasibility of the proposed method but also the output quality of the generated 3D models.

7.2 Geo-based Locality Preserving Key Frame Selection

With the FOV model, each frame corresponds to a camera geo-location and orientation. Here we focus on the frame's location and viewing direction in the geographic domain instead of the traditional pixel domain to extract the most representative frames for 3D reconstruction. Since some UGVs in the candidate area are not recording the target object, we first filter out all the frames that do not contain the target

$$\langle \mathcal{L}_i, \theta_i, \alpha_i \rangle : \| D(\mathcal{L}_i, \boldsymbol{q}) - \theta_i \| \leq \frac{\alpha}{2} \tag{7.1}$$

where $\langle \mathcal{L}_i, \theta_i, \alpha_i \rangle$ is the FOV triplet of the i^{th} frame, $\boldsymbol{q}$ is the geo-location of the target object, and D is a direction function that calculates the viewing direction, given two positions. $\| \cdot \|$ is the angular distance between the two directions. As illustrated in Figure 7.2, the black points are the frames' camera locations from an aerial view. Without loss of generality, we assume that those frames record the object in the center (denoted with a blue square) after the filtering phase. The objective of our algorithm is to select a subset of frames (denoted with red stars), which maintain a minimal, but full, coverage of the target object in the geographic space. In other words, the information loss from any viewing angle

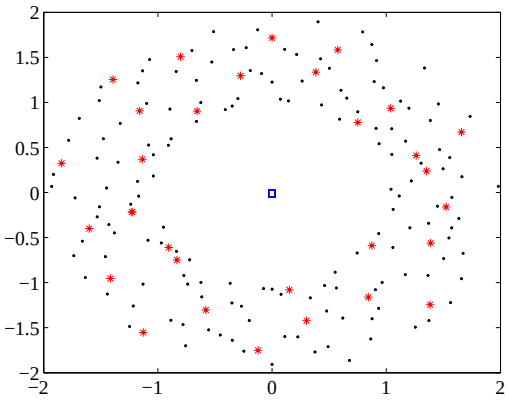


Figure 7.2: Illustration of geo-based active key frame selection algorithm in 2D space. Black points indicate the frames' geo-locations from aerial view. Red stars describe the location of selected key frame subset. Blue square denotes the location of the target object.

towards the target object is minimized by our key frame selection method.

Let $\mathcal{P} = \{\boldsymbol{p}_1, \boldsymbol{p}_2, \ldots, \boldsymbol{p}_n\}$ be the set of all frame geo-location data points (every $\boldsymbol{p}_i$ is equal to $\mathcal{L}_i$ in frames' FOV triplets) and $\mathcal{K} = \{\boldsymbol{k}_1, \boldsymbol{k}_2, \ldots, \boldsymbol{k}_t\} \subset \mathcal{P}$, termed *key location set* in this paper, be the set of the selected location points. Each location data point consists of a coordinate in the World Geodetic System 1984 [18]. Here we propose a coverage gain function in the geographic space, $g(\boldsymbol{p}) = \boldsymbol{w}^T\boldsymbol{p}$, to quantify the target object's viewing angle coverage. Suppose $l = g(\boldsymbol{p}) + \epsilon$ is a real-valued observation from the geographic coverage relation between $\boldsymbol{p}$ and the target object's location $\boldsymbol{q}$, where $\epsilon \sim \mathcal{N}(0, \sigma^2)$ is the measurement error. Thus, the maximum likelihood estimate of w can be obtained by

$$\hat{\boldsymbol{w}} = \arg\min_{\boldsymbol{w}} \sum_{i=1}^{t} (\boldsymbol{w}^T\boldsymbol{k}_i - l_i)^2 \tag{7.2}$$

The key idea of our selection approach is to minimize the difference between the coverage gain based on all frame locations and the *key location set*. Specifically, the average expected square difference of the estimation function g needs to be

minimized. We start by stating the problem formally.

Problem Statement. Given a set of frames $\mathcal{F} = \{f_1, f_2, \ldots, f_n\}$ and their corresponding geo-locations $\mathcal{P} = \{\boldsymbol{p}_1, \boldsymbol{p}_2, \ldots, \boldsymbol{p}_n\}$, we find a key frame subset $\tilde{\mathcal{F}}$ whose corresponding geo-locations are $\mathcal{K} = \{\boldsymbol{k}_1, \boldsymbol{k}_2, \ldots, \boldsymbol{k}_t\} \subset \mathcal{P}$, and the average expected square coverage gain difference, $G_{diff} = \frac{1}{n}\sum_{i=1}^{n} E(l_i - \hat{\boldsymbol{w}}^T \boldsymbol{p})^2$, is minimized.

7.2.1 Heuristic Key Frame Selection

Intuitively, in 2D space of the World Geodetic System (usually an aerial view), for the frames with the same viewing direction towards the target object, we only select the ones in which the target object occupies the largest part of the field-of-view. In other words, from a pixel domain perspective, from the same viewing direction we choose the frame in which the target appears dominantly in the image. This way we can theoretically extract the most diverse viewing directions with a fixed number of frames. However, in a practical implementation, the "*same viewing direction*" needs to be quantified, which might be all the frames within a certain degree range. Thus, this method has difficulty to achieve a globally optimal solution.

The heuristic method is designed for a baseline comparison and it uses a filter-refine paradigm. It first filters out all video frames that do not capture the target based on their θ value in the accompanying sensor data, which is identical to the step we performed in Equation(7.1). In the refinement step, as illustrated in Figure 7.3(a), we equally divide 360 degrees into N directions around the given object $\boldsymbol{q}$ and partition the frame set into N groups based on the camera viewing directions θ. For each group, we select the most geographically covered

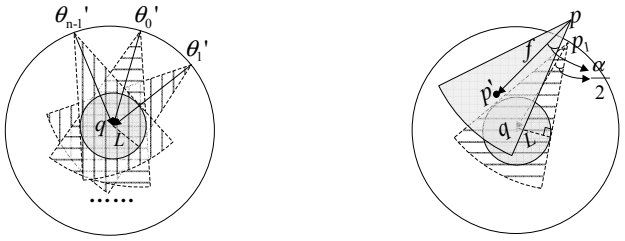


Figure 7.3: (a) Divided direction groups. (b) Computation for the direction-location combined score.

frame $f\langle \boldsymbol{p}, \theta, \alpha \rangle$ determined by a linear combination score of the distance and the direction difference:

$$I_{score}(f, \boldsymbol{q}) = \beta \times \frac{Dist(\boldsymbol{q}, \boldsymbol{p}')}{MaxDist} + (1 - \beta) \times (1 - \cos(\theta'_j, \theta)) \tag{7.3}$$

As shown in Figure 7.3(b), the point $\boldsymbol{p}'$ is obtained with a translation by the Euclidian distance $Dist(\boldsymbol{p}, \boldsymbol{p}') = Dist(\boldsymbol{p}_1, \boldsymbol{q}) = L/\sin(\alpha/2)$ along the viewing direction θ of frame f. $MaxDist$ represents the maximal euclidian distance of pairs of distinct objects in $\mathcal{F}$ for normalization. The cosine $\cos(\theta'_j, \theta)$ is the direction similarity between the group direction θ'_j and the viewing direction θ of f. The tuning parameter β adjusts the balance between the camera location distance and the direction difference. Finally, in each group, the highest scored frame is extracted into the key frame set. We consider frames within a 10-degree range as belonging to the same viewing angle bin; therefore N is set to 36 and β to 0.2 in our experiments.

7.2.2 Adaptive Key Frame Selection

Since the heuristic selection method hardly achieves an optimal solution, we turn to incorporate a manifold structure into reproducing a kernel Hilbert space

to analyze the spatial relationship among the frames. We derive the expected square coverage gain difference as follows:

$$\begin{aligned} G_{diff} &= \frac{1}{n}\sum_{i=1}^{n} E(\hat{\boldsymbol{w}}^T \boldsymbol{p}_i - l_i)^2 \\ &= \frac{1}{n}\sum_{i=1}^{n} \boldsymbol{p}_i{}^T [E(\boldsymbol{w} - \hat{\boldsymbol{w}})(\boldsymbol{w} - \hat{\boldsymbol{w}})^T]\boldsymbol{p}_i \end{aligned} \tag{7.4}$$

By the Gauss-Markov theorem, the covariance matrix of $(\boldsymbol{w} - \hat{\boldsymbol{w}})$ is σ^2 times the inverted Hessian of $\sum_{i=1}^{t}(\boldsymbol{w}^T\boldsymbol{k}_i - l_i)^2$. So G_{diff} can be written as

$$G_{diff} = \sigma^2 + \sigma^2 \mathrm{Tr}(P^T(KK^T)^{-1}P)$$

where $P = [\boldsymbol{p}_1, \ldots, \boldsymbol{p}_n]$ and $K = [\boldsymbol{k}_1, \ldots, \boldsymbol{k}_t]$. We find that the measurement l does not appear in the equation, so the average expected square coverage gain difference only depends on *key location set* $\mathcal{K}$.

This mathematical structure and its semantic objective can be formulated as Transductive Experimental Design (TED), an active learning model from the machine learning community [130]. This problem is often referred to as experiment design in statistics [16] and such an optimization has been verified as being an NP-hard problem [131]. We employ a convex relaxation of the minimization problem proposed by Yu et al. [132]:

$$\min_{\beta, \boldsymbol{\alpha}_i \in \mathbb{R}^n} \sum_{i=1}^{n} \|\boldsymbol{p}_i - K^T\boldsymbol{\alpha}_i\|^2 + \sum_{j=1}^{n} \frac{\alpha_{i,j}^2}{\beta_j} + \gamma\|\boldsymbol{\beta}\|_1$$

where $\boldsymbol{\alpha}_i = [\alpha_{i,1}, \ldots, \alpha_{i,n}]^T$ and $\boldsymbol{\beta} = [\beta_1, \ldots, \beta_n]$ are the auxiliary variables to control the inclusion of examples into the *key location set*. This has been proved to be a convex problem and a global optimal solution is guaranteed. All

candidates with $\beta_j = 0$ can be rejected, since the l_1-norm $\|\boldsymbol{\beta}\|$ enforces a sparse $\boldsymbol{\beta}$.

To achieve the best viewing angle coverage around the target object in geographic space, intuitively we need to take the geometric structure of the data points into consideration. Thus we adopt the Manifold Adaptive Kernel [107] which incorporates the manifold structure into the reproducing kernel Hilbert space (RKHS) to reflect the underlying geometry of the data. To model the structure, we also construct a nearest neighbor graph whose weight matrix elements W_{ij} are 1 if two data points exist within each other's t nearest neighbors [19]. The graph Laplacian accordingly is defined as $L = W' - W$ where W' is given by $W'_{ii} = \sum_j W_{ij}$. Denoting $\mathcal{K}$ as a commonly used kernel such as Gaussian kernel, we obtain the manifold adaptive reproducing kernel as:

$$\mathcal{K}_{\mathcal{M}}(\boldsymbol{p}, \boldsymbol{k}) = \mathcal{K}(\boldsymbol{p}, \boldsymbol{k}) - \lambda \boldsymbol{s}_{\boldsymbol{p}}^T (I + LH)^{-1} L \boldsymbol{s}_{\boldsymbol{k}}$$

where $\boldsymbol{s_p} = (\mathcal{K}(\boldsymbol{p}, \boldsymbol{p_1}), \ldots, \mathcal{K}(\boldsymbol{p}, \boldsymbol{p_n}))$, I is an identity matrix, λ is a constant controlling the smoothness of the functions and H is the kernel matrix in $\mathcal{H}$. $\mathcal{H}$ is a complete Hilbert space of functions $\mathcal{E} \to \mathbb{R}$, where $\mathcal{E}$ is a compact domain in a Euclidean space or a manifold [107]. Cai et al. have shown that this optimization problem can be solved by performing a convex TED in manifold adaptive kernel space [19]. We utilize their model by initializing $\alpha_{i,j} = 1$ and iteratively computing

$$\beta_j = \sqrt{\frac{\sum_{i=1}^{n} \alpha_{i,j}^2}{\gamma}}, \quad j = 1, \ldots, n,$$

$$\boldsymbol{\alpha_i} = (diag(\beta)^{-1} + H)^{-1} \boldsymbol{u_i}, \quad i = 1, \ldots, n,$$

until convergence, where $\boldsymbol{u_i}$ is the i^{th} column vector of H and $H_{ij} = \mathcal{K}(\boldsymbol{p_i}, \boldsymbol{p_j})$. The data points afterwards can be ranked in a descending order with regard to β_j and then selecting the top t as *key location set* $\mathcal{K}$. However, after the convex relaxation of the above problem, an optimal solution is also not guaranteed. Moreover, the locality information can be better formulated into the solution.

7.2.3 Locality Preserving Key Frame Selection

Based on the locality of frames pointing at the same object, each frame location can be linearly reconstructed by its spatial neighboring ones, where the optimal reconstruction coefficients are calculated by [136]:

$$\begin{aligned} &\arg\min_{\boldsymbol{W}} \sum_{i=1}^{n} \|\boldsymbol{p}_i - \sum_{j=1}^{n} \boldsymbol{W}_{ij}\boldsymbol{p}_j\| \\ &\text{s.t.} \sum_{j=1}^{n} \boldsymbol{W}_{ij} = 1, i = 1, \ldots, n \\ &\boldsymbol{W}_{ij} = 0 \quad \text{if} \quad \boldsymbol{p}_j \notin \mathcal{SN}(\boldsymbol{p}_i) \end{aligned} \tag{7.5}$$

where $\boldsymbol{W}_{ij}$ denotes the contribution of the j^{th} frame to construct the i^{th} frame in terms of coverage gain and $\mathcal{SN}(\boldsymbol{p}_i)$ contains the spatial neighbors of the i^{th} location.

To evaluate the representativeness of the selected geo-location, we develop a linear reconstruction approach. The reconstruction error reflects the quality of the selected locations. Let $\{r_1, r_2, \ldots, r_n\}$ be the constructed locations which are determined by minimizing the following cost function:

$$\epsilon(r_1, r_2, \ldots, r_n) = \sum_{i=1}^{t} \|r_{s_i} - \boldsymbol{p}_{s_i}\|^2 + \mu \sum_{i=1}^{n} \|r_i - \sum_{j=1}^{n} \boldsymbol{W}_{ij} r_j\|^2 \tag{7.6}$$

where μ is the regularization parameter, t denotes the number of selected locations, and $\mathcal{S} = \{s_1, s_2, \ldots, s_t\}$ is the set of indices of the selected frames. The first term is the cost function to fix the coordinates of the selected locations. The second term requires that the reconstructed locations share the same local structure with the original ones.

Let $P = [\boldsymbol{p}_1, \boldsymbol{p}_2, \ldots, \boldsymbol{p}_n]$, $R = [r_1, r_2, \ldots, r_n]$ and Γ be an $n \times n$ diagonal matrix whose diagonal entries Γ_{ii} are 1 if $i \in \mathcal{S}$ and 0 otherwise. The above cost function can be reorganized into a matrix form as:

$$\epsilon(R) = tr((R-P)^T \Gamma (R-P)) + \mu tr(R^T M R) \tag{7.7}$$

where $M = (I - \boldsymbol{W})^T (I - \boldsymbol{W})$. To minimize Equation (7.7), we set the gradient of $\epsilon(R)$ to 0 and obtain:

$$\Gamma(R-P) + \mu M R = 0 \tag{7.8}$$

Thus the reconstructed locations are given by:

$$R = (\mu M + \Gamma)^{-1} \Gamma P \tag{7.9}$$

Based on the derived reconstructions, the reconstruction error is measured as:

$$\begin{aligned} \epsilon(\boldsymbol{p}_{s_1}, \boldsymbol{p}_{s_2}, \ldots, \boldsymbol{p}_{s_t}) &= \|P - R\|_F^2 \\ &= \|P - (\mu M + \Gamma)^{-1} \Gamma P\|_F^2 \\ &= \|(\mu M + \Gamma) \mu M P\|_F^2 \end{aligned} \tag{7.10}$$

where $\|\cdot\|_F^2$ is the matrix Frobenius norm.

Minimizing Equation (7.10) is computationally expensive due to its combinatorial nature. To accelerate the learning process, a sequential selection is developed. Denote a set of selected locations as $\{\boldsymbol{p}_{s_1}, \boldsymbol{p}_{s_2}, \ldots, \boldsymbol{p}_{s_{t'}}\}$. Let Λ_i be an $n \times n$ matrix whose ii^{th} entries are 1 and all others are 0. The ${s_{t'+1}}^{th}$ location is determined by solving:

$$s_{t'+1} = \underset{i \notin s_1, s_2, \ldots, s_{t'}}{\arg\min} \|(\mu M + \Gamma + \Lambda_i)^{-1} \mu M P\|_F^2 \tag{7.11}$$

Since matrix M in Equation (7.11) is sparse, we leverage the Sherman-Morrison-Woodbury formula to accelerate the matrix inversion computation [38] and obtain:

$$\begin{aligned} &(\mu M + \Gamma + \Lambda_i)^{-1} = Q - \frac{Q_{*i} Q_{i*}}{1 + Q_{ii}} \\ &\text{and} \quad Q = (\mu M + \Gamma)^{-1} \end{aligned} \tag{7.12}$$

where Q_{*i} and Q_{i*} indicate the i^{th} column and the i^{th} row of Q respectively. Thus the objective function in Equation (7.11) can be written as:

$$\begin{aligned} \|(\mu M + \Gamma + \Lambda_i)^{-1} \mu M P\|_F^2 = \mu^2 tr(Q M P P^T M Q) - \frac{2\mu^2 M P P^T M Q Q_{*i}}{1 + Q_{ii}} \\ + \frac{\mu^2 Q_{i*} Q_{*i} M P P^T M Q_{*i}}{(1 + Q_{ii})^2} \end{aligned} \tag{7.13}$$

Let $A = M P P^T M$, then the optimization problem in Equation (7.11) can

be reorganized as:

$$s_{t'+1} = \underset{i \notin s_1, s_2, \ldots, s_{t'}}{\arg\min} \frac{1}{1+Q_{ii}} \left(\frac{Q_{i*}Q_{*i}Q_{i*}AQ_{*i}}{1+Q_{ii}} - 2Q_{i*}AQQ_{*i} \right) \tag{7.14}$$

7.3 3D Model Reconstruction

We conduct the 3D reconstruction as follows. First, from the frame dataset we extract features with the Scale-Invariant Feature Transform (SIFT) method from the VLFeat library. Afterwards, feature matching and bundle adjustment are performed with the SfM bundler library [109, 110]. Instead of estimating the parameters for all cameras and tracks at once, an incremental approach is employed in this step. The parameters of one single pair of cameras are estimated initially. To avoid the degenerate cases, the pair of images that has the highest matches is chosen. New cameras are added cumulatively and their extrinsic parameters are initialized by the direct linear transform (DLT) technique inside a RANSAC procedure. Meanwhile, the estimate of the intrinsic parameter matrix is provided by DLT as well. This information is later used to initialize the focal length of the new camera. Rather than involving a single camera at a time into the optimization, multiple cameras are added at every increment. When the camera with the greatest number of matches are located, then any camera whose matches number is larger than 75% of the highest matches to the existing 3D points are deemed as one adding batch [109]. For each camera increment, the tracks observed by the new camera are added into the optimization and the sparse bundle adjustment library of Lourakis and Argyros [80] is utilized to minimize the objective function at every iteration. A track is added if other recovered camera observes it, and a well conditioned location estimate

Total # of videos	345
Total # of video frames	77,642
Average length of video (seconds)	55

Table 7.1: Statistics of video dataset.

	Object 1	Object 2	Object 3	Object 4
Average mean	3.961	7.069	11.742	4.663
Standard deviation	0.947	0.637	23.306	0.517
	Object 5	Object 6	Object 7	Object 8
Average mean	5.090	2.438	1.429	4.161
Standard deviation	1.814	0.162	0.381	0.520
	Object 9	Object 10	Object 11	Object 12
Average mean	6.280	13.079	3.196	3.891
Standard deviation	1.937	4.902	0.947	1.231

Table 7.2: The influence to G_{diff} value by choosing different numbers of nearest neighbors.

is given by triangulating. After every run of the optimization, a detection step is followed to remove the outlier tracks that contain any keypoint with a high reprojection error. The optimization is performed again until no more outliers are detected.

Next the output of the SfM step is fed into the Clustering Views for Multi-view Stereo (CMVS) tool to divide the image set into clusters of manageable size and allow them to be processed independently and in parallel [33]. Eventually the Patch-based Multi-view Stereo Software (PMVS2) is executed to produce a set of oriented points instead of a polygonal (or a mesh) model, where both the 3D coordinates and the surface normals are estimated at each oriented point [34].

7.4 Experiments

The main purpose of our key frame selection strategy is to maintain as much view coverage of the target object as possible with the minimally necessary number of frames. Therefore, we focus on two aspects in our experimental evaluation: (a) the geographic coverage gain difference obtained between the original frame set $\mathcal{F}$ and the selected key frame set $\tilde{\mathcal{F}}$, and (b) the processing time reduction achieved for the following 3D reconstruction phase based on these two frame sets with different cardinalities.

In our experiments we utilize the public geo-crowdsourced UGV data from the *GeoVid* app and portal. We retrieved a video dataset as well as its corresponding geo-sensor dataset recorded in two cities, Los Angeles and Singapore. Table 7.1 shows the statistics of the crowdsourced video dataset. Various mobile devices were used for video recording, including the Motorola Milestone, HTC EVO 3D, Samsung Galaxy S4, Asus Transformer and Google Nexus 4. The video resolution is set to 720×480 by the app. We selected 12 target objects (2 in Singapore and 10 in Los Angeles) to which we applied our active key frame selection methods before the 3D reconstruction. Those target objects need to be outdoor and there are relatively large amount of accumulated UGVs around the objects, so that the 3D model reconstruction is feasible. Thumbnails of the selected target objects are illustrated in Figure 7.4.

7.4.1 Geographic Coverage Gain

In order to evaluate whether our proposed algorithm is able to obtain a minimal coverage gain difference, namely a coverage gain close to the one achieved with the whole frame set, we implemented all three key frame selection strategies

Figure 7.4: The sample frames of the selected target objects.

based on geographic analysis for comparison.

In each experiment, we selected at most N (which is 36 described in subsection 7.2.1) frames as a subset (the number of frames may be less due to the absence of coverage from a certain direction in the crowdsrouced UGVs).

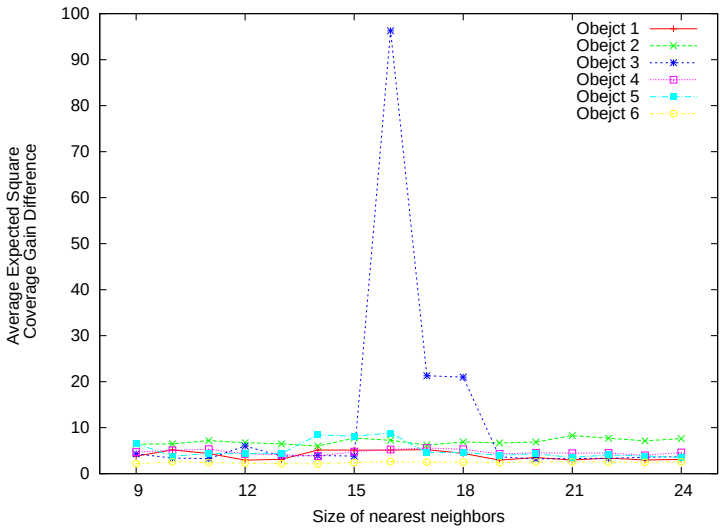


Figure 7.5: Average expected square coverage gain difference on various sizes of nearest neighbors.

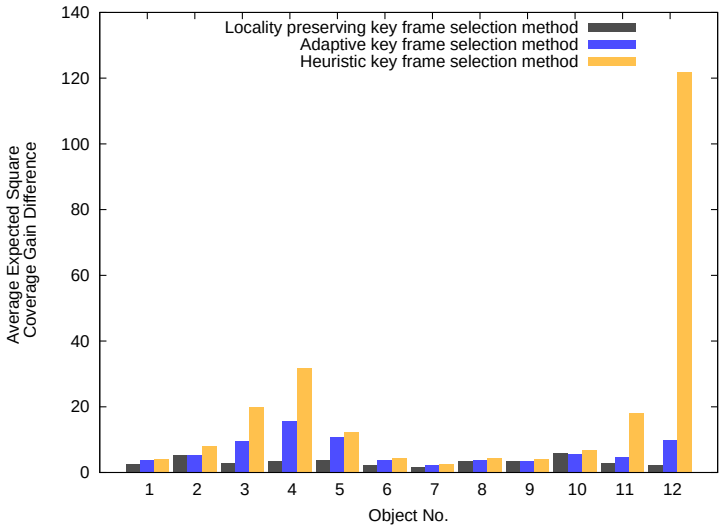
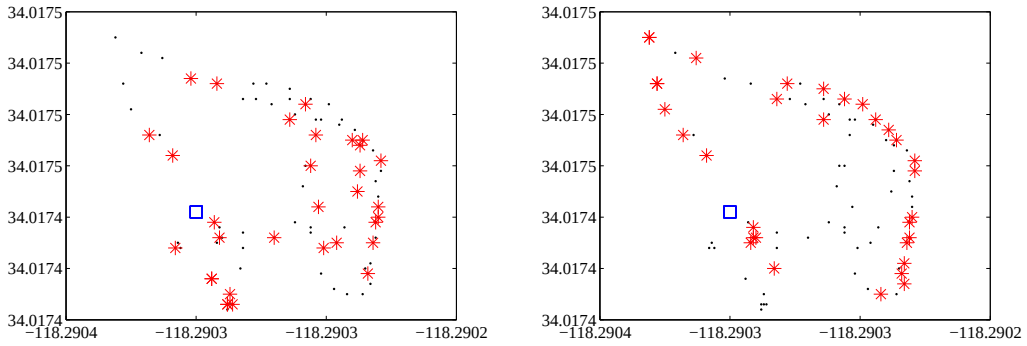


Figure 7.6: Average expected square coverage gain difference of 12 target objects.

We set t constant to ensure that the key frame sets have the same cardinality for all three methods. For the adaptive method, we set all parameters at the same values as tested in MAED [19]. For the locality preserving method, we first evaluate the influence of choosing different nearest neighbor parameters. Table 7.2 reports the average mean and standard deviation of G_{diff} results for

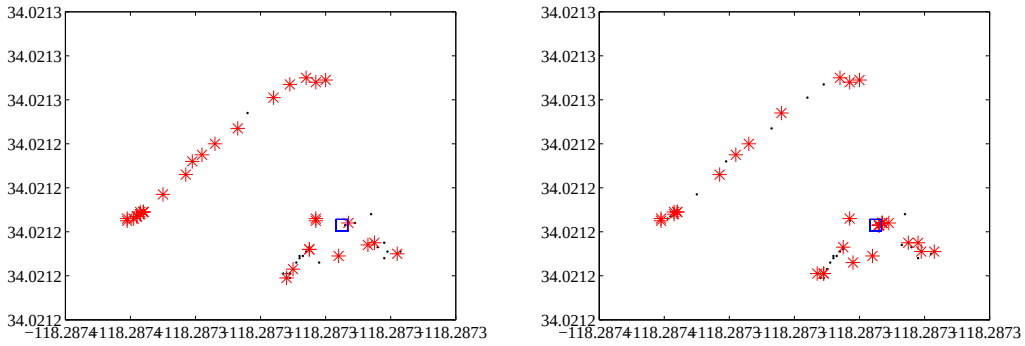


(a) (b)

Figure 7.7: Illustration of key frame selection results with two target objects in aerial view. X- and y-axis denote latitude and longitude. (a) and (b) are the selection results of object No. 1 with locality preserving algorithm and the heuristic method, respectively. Black and red points indicate the geo-locations of $\mathcal{F}$ and $\tilde{\mathcal{F}}$, respectively. The blue square indicates the geo-location of the target object.

all experimental objects by using different numbers of neighbors. We tune this parameter from $t/4$ to $2t/3$ to see its effect to the final result. Except the observable change on Object 3 (see Figure 7.5), the other objects' G_{diff} standard deviation is relatively small (near or less than 1) which indicates the stability of our key frame selection algorithm.

Therefore, we set the number of nearest neighbors as one third of the total candidate frames, $n/3$ and set the regularization parameter μ to 0.01 empirically. Figure 7.6 illustrates the average expected square coverage gain difference, *i.e.*, G_{diff}, calculated between the key frame subset and the whole frame set. The orange bar indicates the difference obtained by the key frames selected with the heuristic method. The blue and black bars are the results of $\tilde{\mathcal{F}}$ extracted by our proposed algorithm, adaptive and locality preserving methods, respectively. Considering all the twelve objects of the experiments, our active selection method consistently achieves less difference, in other words,



(a) (b)

Figure 7.8: Illustration of key frame selection results with two target objects in aerial view. X- and y-axis denote latitude and longitude. (a) and (b) are the selection results of object No. 2 with locality preserving algorithm and the heuristic method, respectively. Black and red points indicate the geo-locations of $\mathcal{F}$ and $\tilde{\mathcal{F}}$, respectively. The blue square indicates the geo-location of the target object.

it is successful in finding a subset that achieves a very close coverage of the target object in geographic space compared to the whole frame set. Moreover, for some objects such as No. 12 and No. 4, the gain difference is notably decreased. In the comparison between the adaptive selection algorithm and the locality preserving selection method, since the locality information is well formulated and included, we find the latter one always performs better (in term of lower coverage gain difference), with the only exception of object No. 10.

Figure 7.7 and 7.8 illustrate two comparison results in detail, between the heuristic method and the locality preserving method. We plot the camera locations of the selected key frames in an aerial view. As illustrated, the key frame set selected by our method includes a wider viewing diversity towards the first object compared with the heuristic method (Figure 7.7(a) and 7.7(b)). For the second object, by contrast, the subsets of the two approaches overlap to a large degree while the result from our method is still slightly better (Figure 7.8(a)

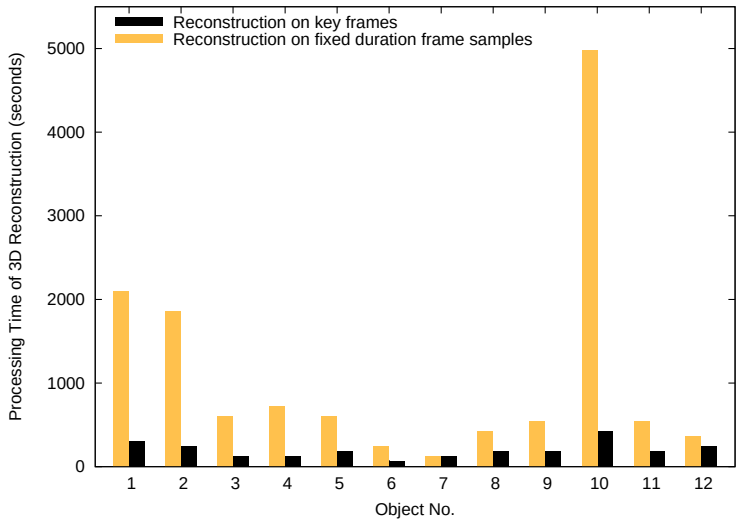


Figure 7.9: Execution time of target object's 3D reconstruction process.

and Figure 7.8(b)). On average, compared with the heuristic approach, our locality preserving key frame selection algorithm decreases the expected square coverage gain difference by 83.14%.

7.4.2 3D Reconstruction Performance

We performed all 3D reconstruction experiments on a 3.4 GHz Intel Core i7-2600 CPU with 4 cores and 8 GB memory. Figure 7.9 illustrates the execution time of the whole 3D reconstruction process for each object. Since the cardinality of the extracted key frame set from all three methods are set to be the same in our experiment, here we only use the key frames extracted by our locality concerned method (which obtains the lowest G_{diff}) for 3D reconstruction comparison. In order to show the efficiency and effectiveness of our active key frame selection method, we sample the collected UGV frames at a fixed duration (1 second in this experiment) as a comparison. The black bar indicates the processing time based on our extracted key frames $\tilde{\mathcal{F}}$, which is significantly less

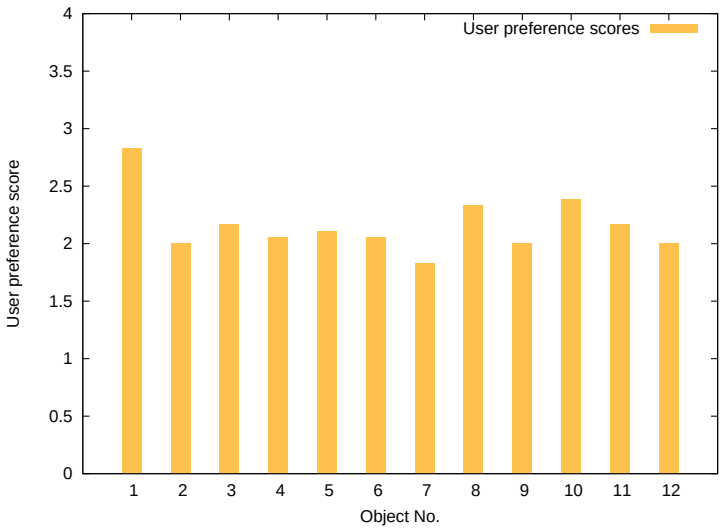
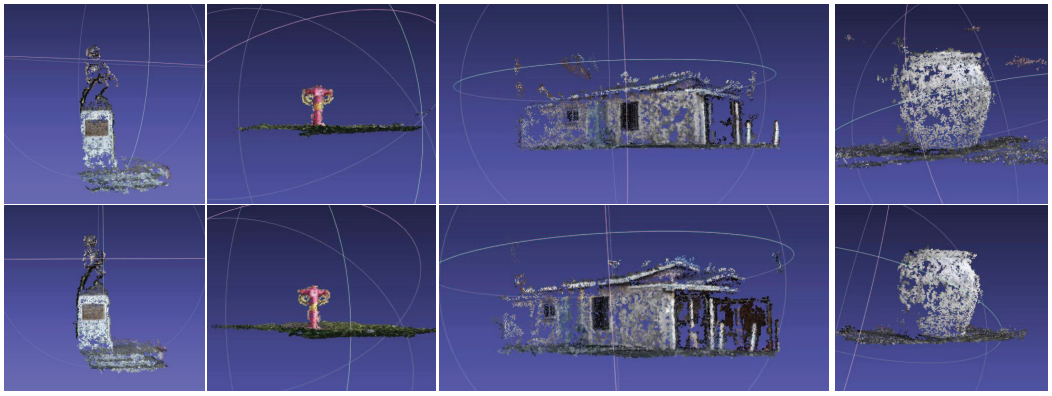


Figure 7.10: Quality comparison between two 3D reconstruction results on two frame sets for 12 target objects.

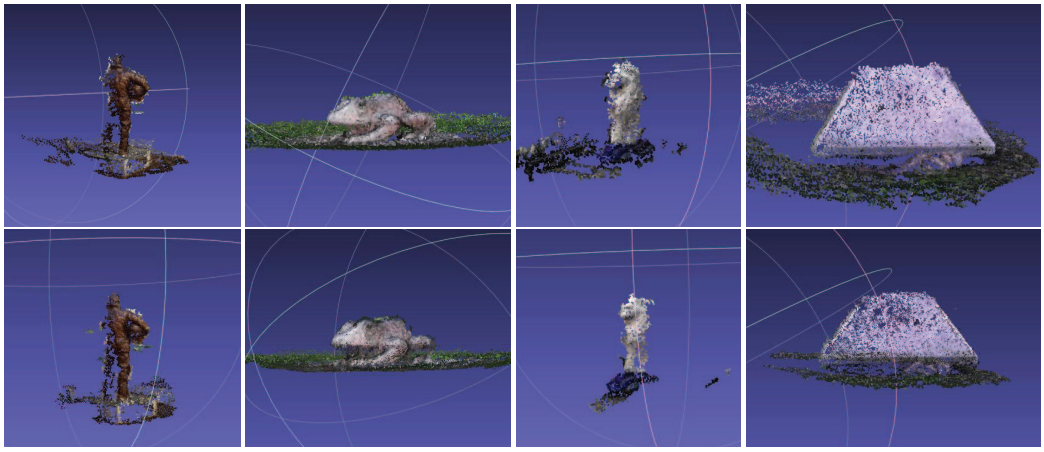
than the processing time based on frames sampled at fixed intervals described in the orange bars. On average, the reconstruction time is shortened by around 15 minutes and the maximal number of frames in $\tilde{\mathcal{F}}$ is only 34.

Since we currently do not have a ground-truth 3D model for the reconstruction evaluation, we conducted a user study to compare the quality of the dense point cloud results based on two frame sets. The participants were requested to carefully examine the 3D scene results visualized via MeshLab[1]. For each target object, two results built from a fixed duration sampled frame set and $\tilde{\mathcal{F}}$ were presented, respectively. After judiciously comparing two 3D scene results, participants were asked to provide marks on the quality for both of them (4 – the quality of the reconstruction result based on $\tilde{\mathcal{F}}$ is much better, 0 – the quality of the reconstruction result based on fixed duration frame samples is much better). Twenty people participated in this study: 14 males and 6 females, including students, engineers, professionals, research staff and faculty.

[1]meshlab.sourceforge.net



(a)



(b)

Figure 7.11: Illustration of 3D reconstruction results of 8 target objects. The top row shows the reconstructed 3D models based on the fixed duration sampled frame set. The bottom row shows the results based on our active key frame selection.

They were asked to consider the completeness of the target object and whether artifacts were visible. We hid the frame set sources of the two results to ensure an unbiased comparison. Figure 7.10 summarizes the results of the user study. As shown, most reconstruction results based on two frame sets were almost the same quality (scores near 2). The key frame set using the active selection method performed slightly worse on objects No. 7 (scores below 2). On the other hand, for some objects such as No. 1, the reconstruction results based on

$\tilde{\mathcal{F}}$ were much better than the other frame set. Some 3D reconstruction results from the two frame sets are shown in Figure 7.11.

7.5 Summary

We presented a novel 3D model reconstruction framework based on the spatial data analysis of the crowdsourced geo-tagged UGVs. As a key component, we leveraged the geospatial properties of those video sources and devised active key frame selection methods upon them. The concept of geographic coverage gain was introduced and the gain difference between the original frames and the key frames was minimized. Our algorithm also incorporates the manifold adaptive kernel and locally linear reconstruction analysis to reflect the underlying geometry. Therefore, the key frame set extracted by our methods maintain the best coverage of the target object in geographic space. The experimental results demonstrate both the effectiveness (averagely the same reconstruction quality) and efficiency (much shorter execution time) of our approach. We illustrate that the coverage difference between the key frames and the overall frame set is reduced. Additionally, the execution time of the 3D reconstruction is shortened by using our selected key frames while the model quality is preserved. Due to pervasive trends and scalability advances in processing contextual data, key frame selection based on geo-sensor data analysis is practical and can complement a content-based approach. In our future work we plan to combine visual features and more sensors to help with frame extraction.

CHAPTER 8

Conclusions and Future Work

8.1 Conclusions

In this dissertation, we examined how to effectively enhance the spatial sensor data accuracy and how to efficiently analyze sensor data to facilitate more versatile and accurate mobile media applications. Several methodologies related to spatial sensor data processing and analysis in the mobile context are proposed. We summarize our contributions as follows:

First, we presented two data correction frameworks for pedestrians-attached and vehicle-attached location sensor data, respectively. We studied the data characteristics and tackled the challenges with linear estimator based on stochastic process and map matching techniques. Our solutions enhanced the location accuracy purely relying on the positioning observations for pedestrians-attached data. As a result, the proposed Eddy system is capable of dynamically selecting

the window size according to the candidate state probability distribution. It outperforms existing approaches on both accuracy and latency aspects.

Second, we presented the design and prototype implementation for camera orientation data correction based on geospatial analysis and image processing techniques. We analyzed the viewable scenes of mobile videos and devised algorithms to estimate more precise orientation data. We demonstrated the effectiveness and efficiency of our methods in experiments.

Afterwards, with more accurate and reliable sensor information obtained, we explore the possibility of applying sensor analysis techniques to new mobile media applications, such as video encoding and 3D model reconstruction. In application part, we first improved the video encoding efficiency based on a sensor-assisted camera motion type characterization framework. Our approach automatically detects video subshot segment boundaries and precisely classifies the motion type of each unit based on a camera movement and shooting direction change analysis. A real-time camera motion characterization demo system was presented as well to show the efficiency advantage of our light-weight sensor data based techniques. We also applied sensor data analysis to simplify the motion estimation in H.264/AVC codec. The search window is decreased and an almost equivalent PSNR performance is obtained. Consequently, the energy computation would be significantly reduced and therefore provide a longer battery life for smartphones with video recording operations.

In the second application, we presented a sensor-assisted UGV-based 3D model reconstruction framework. The system analyzes the spatial sensor data from UGVs to select the most representative key frames as a 3D reconstruction input set. Inspired by the active learning theory, we devised an active key location selection algorithm using a manifold adaptive kernel and locality

preserving reconstruction method. By leveraging the sensor data, our solution provided a key frame set with an improved coverage of the target 3D object from distinct viewing angles in geographic space, but with much fewer frames. In experiments, we showed the significant decrease on the execution time of the whole 3D reconstruction process, while the quality of output 3D models is preserved.

8.2 Future Work

Our research has shown the great potential of leveraging spatial sensor data for mobile media application use. For each proposed work, we listed some applicable future directions can be done to make our system more robust or more adaptable. For example, in video encoding complexity reduction application, we would also look into the utilization of gyroscope which is a new emerging embedded device and has been widely equipped into current mobile phone models. It is capable of measuring the orientation change and suits the motion prediction very well since it is very sensible to a slight movement and the reported relative value is enough for the encoding purpose.

Moreover, there exist several other potential fields that the sensor data analysis could also be applied. We surveyed and plan to extend our research into the location-aware video delivery system. As a result of the pervasiveness of wireless connectivity integrated handheld devices and the rapid deployments of the wireless network technology, streaming multimedia content to mobile peers becomes a popular service that is increasingly available everywhere. Mobile data traffic, according to an annual report from Cisco Systems, continues to grow significantly [47]. The forecast estimates that mobile data traffic will grow

at a CAGR of 61 percent from 2013 to 2018. Moreover, an increasing number of users enjoy the multimedia content in the high-speed vehicular mobility, such as on the public transportation during the daily commute or travelling. The network condition, however, is not always stable along the whole journey of the media content consuming trip. A number of studies have reported the significant bandwidth variation over different geo-locations. Even within the same area/cell site, the bandwidth may vary due to factors like the surrounding environment and the time of day. One typical situation is that a user is watching an online video in a fast-moving train, whose location is continuously changing. The streaming service in this case may be effected or even disrupted due to the perceptible bandwidth disparity. Meanwhile, it is extremely difficult for providers to eliminate bandwidth variation across the entire service area in geographic space.

Recently attention has focused on the Dynamic Adaptive Streaming over HTTP (DASH) standard. Its main features consist of (a) splitting a large video file into segments, (b) providing client-initiated flexible bandwidth adaptation by enabling stream switching among differently encoded segments. Building on this technique, we plan to investigate a smart media delivery system, with the novel feature of future bandwidth prediction for mobile devices, in order to deal with such available bandwidth variation phenomenon. Inspired by the correlation, explored by several studies [98, 129, 42], between geospatial space and bandwidth dimension, we plan to fuse the bandwidth map gathering functionality into our current community-driven spatial sensor data crowdsourced platform. It will enable the near-future bandwidth availability estimation within an accepted accuracy, and a media streaming system with quality adaptation taking future bandwidth estimation into consideration.

Bibliography

[1] F. Ahammed, J. Taheri, A. Zomaya, and M. Ott. VLOCI2: Improving 2D Location Coordinates using Distance Measurements in GPS-Equipped VANETs. In *14th ACM International Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems*, 2011. 5

[2] M. T. Ahmed, M. N. Dailey, J. L. Landabaso, and N. Herrero. Robust Key Frame Extraction for 3D Reconstruction from Video Streams. In *International Conference on Computer Vision Theory and Applications*, pages 231–236, 2010. 25

[3] H. Alt, A. Efrat, G. Rote, and C. Wenk. Matching Planar Maps. In *14th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 589–598. Society for Industrial and Applied Mathematics, 2003. 16

[4] H. Alt and M. Godau. Computing the fréchet distance between two polygonal curves. *International Journal of Computational Geometry & Applications*, 5(01n02):75–91, 1995. 16

[5] E. Ardizzone, M. La Cascia, A. Avanzato, and A. Bruna. Video Indexing Using MPEG Motion Compensation Vectors. In *IEEE International Conference on Multimedia Computing and Systems*, volume 2, pages 725–729, July 1999. 22, 23

[6] S. Arslan Ay, S. H. Kim, and R. Zimmermann. Relevance Ranking in Georeferenced Video Search. *Multimedia Systems Journal*, pages 105–125, 2010. 15

[7] S. Arslan Ay, R. Zimmermann, and S. Kim. Viewable Scene Modeling for Geospatial Video Search. In *16th ACM International Conference on Multimedia*, pages 309–318, 2008. 4

[8] S. Arslan Ay, R. Zimmermann, and S. H. Kim. Viewable Scene Modeling for Geospatial Video Search. In *16th ACM International Conference on Multimedia*, pages 309–318, 2008. 15, 29

[9] P. T. Baker and Y. Aloimonos. Calibration of A Multicamera Network. In *IEEE Computer Vision and Pattern Recognition Workshop*, volume 7, pages 72–72, 2003. 21

[10] S. Battiato, G. Gallo, G. Puglisi, and S. Scellato. SIFT Features Tracking for Video Stabilization. In *14th International Conference on Image Analysis and Processing*, pages 825–830, Sept. 2007. 24

[11] S. Bell, W. Jung, and V. Krishnakumar. WiFi-based Enhanced Positioning Systems: Accuracy through Mapping, Calibration, and Classification. In *2nd ACM SIGSPATIAL International Workshop on Indoor Spatial Awareness*, 2010. 5, 19

[12] D. Bernstein and A. Kornhauser. An Introduction to Map Matching for Personal Navigation Assistants. 1998. 15

[13] R. Billen, E. Joao, and D. Forrest. *Dynamic and Mobile GIS: Investigating Changes in Space and Time*. CRC Press, 2006. 17, 18

[14] J. Bloit and X. Rodet. Short-time Viterbi for Online HMM Decoding: Evaluation on A Real-time Phone Recognition Task. In *IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 2121–2124, 2008. 51

[15] P. Bouthemy, M. Gelgon, and F. Ganansia. A Unified Approach to Shot Change Detection and Camera Motion Characterization. *IEEE Transaction on Circuits and Systems for Video Technology*, 9(7):1030–1044, Oct. 1999. 22

[16] S. P. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004. 127

[17] S. Brakatsoulas, D. Pfoser, R. Salas, and C. Wenk. On Map-Matching Vehicle Tracking Data. In *31st International Conference on Very Large Data Bases*, pages 853–864, 2005. 16

[18] R. K. Burkhard. *Geodesy for the Layman*. US Department of Commerce, National Oceanic and Atmospheric Administration, 1985. 124

[19] D. Cai and X. He. Manifold Adaptive Experimental Design for Text Categorization. *IEEE Transactions on Knowledge and Data Engineering*, 24(4):707–719, 2012. 128, 135

[20] S. S. Chawathe. Segment-based Map Matching. In *IEEE Intelligent Vehicles Symposium*, pages 1190–1197, 2007. 4

[21] X. Chen, Z. Zhao, A. Rahmati, Y. Wang, and L. Zhong. SaVE: Sensor-assisted Motion Estimation for Efficient H.264/AVC Video Encoding. In *17th ACM International Conference on Multimedia*, pages 381–390, 2009. 4

[22] X. Chen, Z. Zhao, A. Rahmati, Y. Wang, and L. Zhong. SaVE: Sensor-assisted Motion Estimation for Efficient H.264/AVC Video Encoding. In *17th ACM International Conference on Multimedia*, pages 381–390, 2009. 24

[23] A. R. Chowdhury, R. Chellappa, S. Krishnamurthy, and T. Vo. 3D Face Reconstruction from Video Using a Generic Model. In *IEEE International Conference on Multimedia and Expo*, volume 1, pages 449–452, 2002. 25

[24] I. Constandache, S. Gaonkar, M. Sayler, R. Choudhury, and L. Cox. EnLoc: Energy-Efficient Localization for Mobile Phones. In *31st IEEE International Conference on Computer Communications*, pages 2716–2720, 2009. 5, 19

[25] J. Denzler, V. Schless, D. Paulus, and H. Niemann. Statistical Approach to Classification of Flow Patterns for Motion Detection. In *International Conference on Image Processing*, pages 517–520, 1996. 22

[26] S. Divvala, D. Hoiem, J. Hays, A. Efros, and M. Hebert. An Empirical Study of Context in Object Detection. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2009. 3

[27] Z. Dong, G. Zhang, J. Jia, and H. Bao. Keyframe-based real-time camera tracking. In *12th International Conference on Computer Vision*, pages 1538–1545. IEEE, 2009. 26

[28] L. Duan, J. Jin, Q. Tian, and C. Xu. Nonparametric motion characterization for robust classification of camera motion patterns. *IEEE Transaction on Multimedia*, 8(2):323–340, 2006. 23

[29] B. Epshtein, E. Ofek, Y. Wexler, and P. Zhang. Hierarchical Photo Organization Using Geo-relevance. In *15th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, 2007. 4

[30] R. Ewerth, M. Schwalb, P. Tessmann, and B. Freisleben. Estimation of Arbitrary Camera Motion in MPEG Videos. In *17th International Conference on Pattern Recognition*, volume 1, pages 512–515, Aug. 2004. 22, 23

[31] S. Fang and R. Zimmermann. Enacq: Energy-efficient GPS Trajectory Data Acquisition based on Improved Map Matching. In *19th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, pages 221–230, 2011. 42, 60, 67

[32] M. A. Fischler and R. C. Bolles. Random Sample Consensus: a Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography. *Communications of the ACM*, pages 381–395, 1981. 92

[33] Y. Furukawa, B. Curless, S. M. Seitz, and R. Szeliski. Towards Internet-scale Multi-view Stereo. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1434–1441, 2010. 25, 97, 133

[34] Y. Furukawa and J. Ponce. Accurate, Dense, and Robust Multiview Stereopsis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 1362–1376, 2010. 98, 133

[35] Y. Gao, J. Tang, R. Hong, Q. Dai, T. Chua, and R. Jain. W2Go: a Travel Guidance System by Automatic Landmark Ranking. In *18th ACM International Conference on Multimedia*, pages 123–132, 2010. 4

[36] M. Goesele, N. Snavely, B. Curless, H. Hoppe, and S. M. Seitz. Multi-view Stereo for Community Photo Collections. In *International Conference on Computer Vision*, pages 1–8, 2007. 25

[37] C. Y. Goh, J. Dauwels, N. Mitrovic, M. Asif, A. Oran, and P. Jaillet. Online Map-matching based on Hidden Markov Model for Real-time Traffic Sensing Applications. In *15th IEEE International Conference on Intelligent Transportation Systems*, pages 776–781, 2012. 19, 51, 65

[38] G. H. Golub and C. F. Van Loan. *Matrix Computations*, volume 3. JHU Press, 2012. 131

[39] J. S. Greenfeld. Matching GPS Observations to Locations on A Digital Map. In *Transportation Research Board 81st Annual Meeting*, 2002. 15, 17

[40] A. Gros, A. Goldwurm, M. Cadolle-Bel, P. Goldoni, J. Rodriguez, L. Foschini, M. Del Santo, and P. Blay. The INTEGRAL IBIS/ISGRI System Point Spread Function and Source Location Accuracy. *Arxiv preprint astro-ph/0311176*, 2003. 5

[41] J. Hao, G. Wang, B. Seo, and R. Zimmermann. Keyframe Presentation for Browsing of User-generated Videos on Map Interfaces. In *19th ACM International Conference on Multimedia*, pages 1013–1016, 2011. 4

[42] J. Hao, R. Zimmermann, and H. Ma. GTube: Geo-Predictive Video Streaming over HTTP in Mobile Environments. In *5th ACM Multimedia Systems Conference*, 2014. 146

[43] J. Heuer and A. Kaup. Global Motion Estimation in Image Sequences Using Robust Motion Vector Field Segmentation. In *7th ACM International conference on Multimedia*, pages 261–264, 1999. 22, 23

[44] P. Hii and A. Zaslavsky. Improving Location Accuracy by Combining WLAN Positioning and Sensor Technology. In *1st Workshop on REALWSN*, 2005. 5, 19

[45] G. Hong, A. Rahmati, Y. Wang, and L. Zhong. SenseCoding: Accelerometer-assisted Motion Estimation for Efficient Video Encoding. In *16th ACM International Conference on Multimedia*, pages 749–752, 2008. 24

[46] T.-H. Hwang, K.-H. Choi, I.-H. Joo, and J.-H. Lee. MPEG-7 Metadata for Video-based GIS Applications. In *Geoscience and Remote Sensing Symposium*, pages 3641–3643, 2003. 14

[47] Index, Cisco Visual Networking. Global Mobile Data Traffic Forecast Update, 2013–2018, Cisco White Paper, Feb. 5, 2014. 145

[48] A. Irschara, C. Zach, J.-M. Frahm, and H. Bischof. From Structure-from-Motion Point Clouds to Fast Location Recognition. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 2599–2606, 2009. 21

[49] J. Jannotti and J. Mao. Distributed calibration of smart cameras. In *Workshop on Distributed Smart Cameras*, 2006. 21

[50] R. Jin, Y. Qi, and A. Hauptmann. A Probabilistic Model for Camera Zoom Detection. In *16th International Conference on Pattern Recognition*, volume 3, pages 859–862, 2002. 22

[51] K. Jinzenji, S. Ishibashi, and H. Kotera. Algorithm for Automatically Producing Layered Sprites by Detecting Camera Movement. In *Intl. Conference on Image Processing*, volume 1, pages 767–770, Oct. 1997. 22

[52] T. Judd, K. Ehinger, F. Durand, and A. Torralba. Learning to Predict Where Humans Look. In *12th International Conference on Computer Vision*, pages 2106–2113, 2009. 86

[53] D. Jwo, M. Chen, C. Tseng, and T. Cho. Adaptive and Nonlinear Kalman Filtering for GPS Navigation Processing. *Kalman Filter: Recent Advances and Applications*, 2009. 19

[54] L. Kaminski, R. Kowalik, Z. Lubniewski, and A. Stepnowski. "VOICE MAPS" - Portable, Dedicated GIS for Supporting the Street Navigation and Self-dependent Movement of the Blind. In *2nd International Conference on Information Technology*, pages 153–156, 2010. 4

[55] A. R. Karlin, M. S. Manasse, L. Rudolph, and D. D. Sleator. Competitive Snoopy Caching. *Algorithmica*, 3(1-4):79–119, 1988. 52, 56

[56] T. Kato, Y. Terada, M. Kinoshita, H. Kakimoto, H. Isshiki, M. Matsuishi, A. Yokoyama, and T. Tanno. Real-time Observation of Tsunami by RTK-GPS. *Earth Planets And Space*, 52(10):841–846, 2000. 4

[57] C. Kee and B. Parkinson. Wide Area Differential GPS as A Future Navigation System in The US. In *IEEE Position Location and Navigation Symposium*, 1994. 4

[58] L. Kennedy and M. Naaman. Generating Diverse and Representative Image Search Results for Landmarks. In *17th International World Wide Web Conferences*, pages 297–306, 2008. 4

[59] J. Kim, H. Chang, J. Kim, and H. Kim. Efficient Camera Motion Characterization for MPEG Video Indexing. In *IEEE International Conference on Multimedia and Expo*, pages 1171–1174, 2000. 22, 23

[60] K.-H. Kim, S.-S. Kim, S.-H. Lee, J.-H. Park, and J.-H. Lee. The Interactive Geographic Video. In *Geoscience and Remote Sensing Symposium*, pages 59–61, 2003. 14

[61] S. H. Kim, Y. Lu, G. Constantinou, C. Shahabi, G. Wang, and R. Zimmermann. MediaQ: Mobile Multimedia Management System. In *5th ACM Multimedia Systems Conference*, pages 224–235, 2014. 28

[62] W. Kim, G.-I. Jee, and J. Lee. Efficient Use of Digital Road Map in Various Positioning for ITS. In *IEEE Position Location and Navigation Symposium*, pages 170–176, 2000. 17, 18

[63] G. Klein and D. Murray. Parallel Tracking and Mapping for Small AR Workspaces. In *6th IEEE International Symposium on Mixed and Augmented Reality*, pages 225–234, 2007. 26

[64] M. Kroepfl, Y. Wexler, and E. Ofek. Efficiently Locating Photographs in Many Panoramas. In *18th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, pages 119–128, 2010. 20

[65] J. LaMance, J. DeSalas, and J. Jarvinen. Assisted GPS: A Low-Infrastructure Approach. *GPS World*, 13, 2002. 33

[66] A. LaMarca, Y. Chawathe, S. Consolvo, J. Hightower, I. Smith, J. Scott, T. Sohn, J. Howard, J. Hughes, F. Potter, et al. Place Lab: Device Positioning Using Radio Beacons in the Wild. In *Pervasive Computing*, pages 116–133. Springer, 2005. 47

[67] K. C. Lee, W.-C. Lee, and H. V. Leong. Nearest Surrounder Queries. *IEEE Transactions on Knowledge and Data Engineering*, pages 1444–1458, 2010. 82

[68] T. Lertrusdachakul, T. Aoki, and H. Yasuda. Camera Motion Estimation by Image Feature Analysis. *Pattern Recognition and Image Analysis*, pages 618–625, 2005. 24

[69] M. Lhuillier and L. Quan. A Quasi-dense Approach to Surface Reconstruction from Uncalibrated Images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(3):418–433, 2005. 25

[70] X. Li, C. Wu, C. Zach, S. Lazebnik, and J.-M. Frahm. Modeling and Recognition of Landmark Image Collections Using Iconic Scene Graphs. In *European Conference on Computer Vision*, pages 427–440. 2008. 21

[71] Y. Li, N. Snavely, and D. P. Huttenlocher. Location Recognition Using Prioritized Feature Matching. In *European Conference on Computer Vision*, pages 791–804. 2010. 21

[72] H.-H. Liao, Y. Lin, and G. Medioni. Aerial 3D Reconstruction with Line-constrained Dynamic Programming. In *International Conference on Computer Vision*, pages 1855–1862, 2011. 25

[73] L. Liao, D. J. Patterson, D. Fox, and H. Kautz. Learning and Inferring Transportation Routines. *Artificial Intelligence*, 171(5):311–331, 2007. 17

[74] K. Lin, A. Kansal, D. Lymberopoulos, and F. Zhao. Energy-accuracy Aware Localization for Mobile Devices. *ACM International Conference on Mobile Systems*, 2010. 34

[75] L. Ling, I. S. Burrent, and E. Cheng. A Dense 3D Reconstruction Approach from Uncalibrated Video Sequences. In *IEEE International Conference on Multimedia and Expo Workshops*, pages 587–592, 2012. 25

[76] H. Liu, T. Mei, J. Luo, H. Li, and S. Li. Finding Perfect Rendezvous on the Go: Accurate Mobile Visual Localization and Its Applications to Routing. In *20th ACM International Conference on Multimedia*, pages 9–18, 2012. 21

[77] X. Liu, M. Corner, and P. Shenoy. SEVA: Sensor-Enhanced Video Annotation. In *13th ACM International Conference on Multimedia*, pages 618–627, 2005. 14

[78] Z. Lotker, B. Patt-Shamir, and D. Rawitz. Rent, Lease or Buy: Randomized Algorithms for Multislope Ski Rental. *SIAM Journal on Discrete Mathematics*, 26(2):718–736, 2012. 52

[79] Y. Lou, C. Zhang, Y. Zheng, X. Xie, W. Wang, and Y. Huang. Map-matching for Low-sampling-rate GPS Trajectories. In *17th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, pages 352–361, 2009. 42, 51

[80] M. Lourakis and A. Argyros. The Design and Implementation of a Generic Sparse Bundle Adjustment Software Package based on the Levenberg-Marquardt Algorithm. Technical report, Technical Report 340, Institute of Computer Science-FORTH, Heraklion, Crete, Greece, 2004. 132

[81] D. Lowe. Distinctive Image Features from Scale-Invariant Keypoints. *International Journal of Computer Vision*, pages 91–110, 2004. 92

[82] X. Lu, C. Wang, J. Yang, Y. Pang, and L. Zhang. Photo2Trip: Generating Travel Routes from Geo-tagged Photos for Trip Planning. In *18th ACM International Conference on Multimedia*, pages 143–152, 2010. 4

[83] Z. Luo, H. Li, J. Tang, R. Hong, and T.-S. Chua. ViewFocus: Explore Places of Interests on Google Maps Using Photos with View Direction Filtering. In *17th ACM International Conference on Multimedia*, pages 963–964, 2009. 20

[84] H. Ma, R. Zimmermann, and S. H. Kim. HUGVid: Handling, Indexing and Querying of Uncertain Geo-tagged Videos. In *20th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, pages 319–328, 2012. 4, 86

[85] J. W. Mills, A. Curtis, B. Kennedy, S. W. Kennedy, and J. D. Edwards. Geospatial Video for Field Data Collection. *Applied Geography*, 30(4):533–547, 2010. 4

[86] A. Mohamed and K. Schwarz. Adaptive Kalman Filtering for INS/GPS. *Journal of Geodesy*, 73(4):193–203, 1999. 46

[87] L. Monteiro, T. Moore, and C. Hill. What is The Accuracy of DGPS? *Journal of Navigation*, 58, 2005. 4

[88] P. Mordohai, J.-M. Frahm, A. Akbarzadeh, B. Clipp, C. Engels, D. Gallup, P. Merrell, C. Salmi, S. Sinha, B. Talton, et al. Real-time Video-based Reconstruction of Urban Environments. *ISPRS Working Group*, 2007. 27

[89] E. Mouragnon, M. Lhuillier, M. Dhome, F. Dekeyser, and P. Sayd. Real Time Localization and 3D Reconstruction. In *IEEE Conference on Computer Vision and Pattern Recognition*, volume 1, pages 363–370, 2006. 25, 26

[90] P. Newson and J. Krumm. Hidden Markov Map Matching Through Noise and Sparseness. In *17th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, pages 336–343, 2009. 17, 18, 42, 47, 66

[91] D. Nistér. *Automatic Dense Reconstruction from Uncalibrated Video Sequences*. PhD thesis, KTH, 2001. 25

[92] V. Otsason, A. Varshavsky, A. LaMarca, and E. De Lara. Accurate GSM Indoor Localization. *7th International Conference on Ubiquitous Computing*, 2005. 19

[93] M. Park, J. Luo, R. T. Collins, and Y. Liu. Beyond GPS: Determining the Camera Viewing Direction of a Geotagged Image. In *18th ACM International Conference on Multimedia*, pages 631–634, 2010. 20

[94] O. Pink and B. Hummel. A Statistical Approach to Map Matching Using Road Network Geometry, Topology and Vehicular Motion Constraints. In *Intelligent Transportation Systems*, pages 862–867, 2008. 17, 18

[95] M. Pollefeys, L. Van Gool, M. Vergauwen, F. Verbiest, K. Cornelis, J. Tops, and R. Koch. Visual Modeling with A Hand-held Camera. *International Journal of Computer Vision*, 59(3):207–232, 2004. 26

[96] M. A. Quddus, W. Y. Ochieng, and R. B. Noland. Current Map-Matching Algorithms for Transport Applications: State-of-the Art and Future Research Directions. *Transportation Research Part C: Emerging Technologies*, 15(5):312–328, 2007. 17

[97] M. A. Quddus, W. Y. Ochieng, L. Zhao, and R. B. Noland. A General Map Matching Algorithm for Transport Telematics Applications. *GPS Solutions*, 7(3):157–167, 2003. 17

[98] H. Riiser, T. Endestad, P. Vigmostad, C. Griwodz, and P. Halvorsen. Video Streaming Using a Location-Based Bandwidth-Lookup Service for Bitrate Planning. *ACM Transactions on Multimedia Computing, Communications and Applications*, 2012. 146

[99] J. Sasiadek, Q. Wang, and M. Zeremba. Fuzzy Adaptive Kalman Filtering for INS/GPS Data Fusion. In *IEEE International Symposium on Intelligent Control*, 2000. 19

[100] T. Sattler, B. Leibe, and L. Kobbelt. Fast Image-based Localization Using Direct 2D-to-3D Matching. In *IEEE International Conference on Computer Vision*, pages 667–674, 2011. 21

[101] B. Seo, J. Hao, and G. Wang. Sensor-rich Video Exploration on a Map Interface. In *19h ACM International conference on Multimedia*, 2011. 28

[102] J. K. Seo, S. H. Kim, C. W. Jho, and H. K. Hong. 3D Estimation and Key-Frame Selection for Match Move. In *International Technical Conference on Circuits Systems, Computers and Communications*, pages 1282–1285, 2003. 25

[103] Y.-H. Seo, S.-H. Kim, K.-S. Doo, and J.-S. Choi. Optimal Keyframe Selection Algorithm for Three-dimensional Reconstruction in Uncalibrated Multiple Images. *Optical Engineering*, 47(5), 2008. 25

[104] Z. Shen, S. Arslan Ay, S. H. Kim, and R. Zimmermann. Automatic Tag Generation and Ranking for Sensor-rich Outdoor Videos. In *19th ACM International Conference on Multimedia*, pages 93–102, 2011. 4, 82, 100

[105] J. Shi and C. Tomasi. Good Features to Track. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 593–600, 1994. 90

[106] H.-Y. Shum, Q. Ke, and Z. Zhang. Efficient bundle adjustment with virtual key frames: A hierarchical approach to multi-frame structure from motion. In *IEEE Conference on Computer Vision and Pattern Recognition*, volume 2, 1999. 25

[107] V. Sindhwani, P. Niyogi, and M. Belkin. Beyond the Point Cloud: from Transductive to Semi-supervised Learning. In *International Conference on Machine Learning*, pages 824–831, 2005. 128

[108] M. Slaney. Web-scale multimedia analysis: does content matter? *IEEE Multimedia*, 18(2):12–15, 2011. 3

[109] N. Snavely, S. M. Seitz, and R. Szeliski. Photo Tourism: Exploring Photo Collections in 3D. In *ACM Transactions on Graphics*, volume 25, pages 835–846, 2006. 4, 97, 132

[110] N. Snavely, S. M. Seitz, and R. Szeliski. Modeling the World from Internet Photo Collections. *International Journal of Computer Vision*, pages 189–210, 2008. 132

[111] R. Šrámek, B. Brejová, and T. Vinař. On-line Viterbi Algorithm and Its Relationship to Random Walks. *arXiv:0704.0062*, 2007. 51

[112] S. Steiniger, M. Neun, and A. Edwardes. Foundations of Location Based Services. *Lecture Notes on LBS*, 1:272, 2006. 2

[113] M. Sturza. GPS Navigation using Three Satellites and A Precise Clock. *NAVIGATION: Journal of the Institute of Navigation*, 30, 1983. 4

[114] I. Suveg and G. Vosselman. 3D reconstruction of Building Models. *International Archives of Photogrammetry and Remote Sensing*, 33(B2; PART 2):538–545, 2000. 8

[115] R. Szeliski. Image Alignment and Stitching: a Tutorial. *Foundations and Trends in Computer Graphics and Vision*, 2006. 98

[116] A. Thiagarajan, L. Ravindranath, H. Balakrishnan, S. Madden, L. Girod, et al. Accurate, Low-Energy Trajectory Mapping for Mobile Devices. In *8th USENIX Conference on Networked Systems Design and Implementation*, pages 20–33, 2011. 42

[117] A. Thiagarajan, L. Ravindranath, K. LaCurts, S. Madden, H. Balakrishnan, S. Toledo, and J. Eriksson. VTrack: Accurate, Energy-Aware Road Traffic Delay Estimation Using Mobile Phones. In *7th ACM Conference on Embedded Networked Sensor Systems*, pages 85–98, 2009. 17, 18

[118] C. Torniai, S. Battle, and S. Cayzer. Sharing, Discovering and Browsing Geotagged Pictures on the World Wide Web. *The Geospatial Web, Advanced Information and Knowledge Processing*, 1:159–170, 2007. 4

[119] P. Torr, A. W. Fitzgibbon, and A. Zisserman. Maintaining Multiple Motion Model Hypotheses over Many Views to Recover Matching and Structure. In *6th International Conference on Computer Vision*, pages 485–491. IEEE, 1998. 26

[120] P. H. Torr. Geometric Motion Segmentation and Model Selection. *Philosophical Transactions of the Royal Society of London. Series A: Mathematical, Physical and Engineering Sciences*, pages 1321–1340, 1998. 25

[121] M. Ulrich and S. Martin. Sensor Assited Video Compression. European Patent Application EP1921867. 25

[122] A. J. Viterbi. Error Bounds for Convolutional Codes and an Asymptotically Optimum Decoding Algorithm. *IEEE Transactions on Information Theory*, pages 260–269, 1967. 42

[123] G. Wang, B. Seo, Y. Yin, R. Zimmermann, and Z. Shen. Oscor: An Orientation Sensor Data Correction System for Mobile Generated Contents. In *21st ACM International Conference on Multimedia*, pages 439–440, 2013. 99

[124] G. Wang, B. Seo, and R. Zimmermann. Automatic Positioning Data Correction for Sensor-annotated Mobile Videos. In *20th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, pages 470–473, 2012. 85

[125] G. Wang, B. Seo, and R. Zimmermann. Motch: an Automatic Motion Type Characterization System for Sensor-rich Videos. In *20th ACM International Conference on Multimedia*, pages 1319–1320, 2012. 116

[126] R. Wang and T. Huang. Fast Camera Motion Analysis in MPEG Domain. In *International Conference on Image Processing*, volume 3, pages 691–694, 1999. 22, 23

[127] Z. Wang, L. Sun, and S. Yang. Efficient Relative Camera Orientation Detection for Mobile Applications. In *1st ACM International Workshop on Mobile Location-based Service*, pages 53–62, 2011. 21

[128] C. E. White, D. Bernstein, and A. L. Kornhauser. Some Map Matching Algorithms for Personal Navigation Assistants. *Transportation Research Part C: Emerging Technologies*, 8(1):91–108, 2000. 15, 16

[129] J. Yao, S. S. Kanhere, and M. Hassan. Improving QoS in High-Speed Mobility Using Bandwidth Maps. *IEEE Transaction on Mobile Computing*, 2012. 146

[130] K. Yu, J. Bi, and V. Tresp. Transductive Experiment Design. 2005. 127

[131] K. Yu, J. Bi, and V. Tresp. Active Learning via Transductive Experimental Design. In *International Conference on Machine Learning*, pages 1081–1088, 2006. 127

[132] K. Yu, S. Zhu, W. Xu, and Y. Gong. Non-greedy Active Learning for Text Categorization Using Convex Ansductive Experimental Design. In *ACM SIGIR Conference*, pages 635–642, 2008. 127

[133] J. Yuan, Y. Zheng, C. Zhang, X. Xie, and G.-Z. Sun. An Interactive-voting based Map Matching Algorithm. In *11th IEEE International Conference on Mobile Data Management*, pages 43–52, 2010. 42, 51

[134] P. Zandbergen. Accuracy of iPhone Locations: A Comparison of Assisted GPS, WiFi and Cellular Positioning. *Transactions in GIS*, 13, 2009. 33

[135] H. Zhang, B. Li, and D. Yang. Keyframe Detection for Appearance-based Visual SLAM. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2071–2076, 2010. 26

[136] L. Zhang, C. Chen, J. Bu, D. Cai, X. He, and T. S. Huang. Active Learning Based on Locally Linear Reconstruction. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(10):2026–2038, 2011. 129

[137] Y. Zhang, G. Wang, B. Seo, and R. Zimmermann. Multi-video Summary and Skim Generation of Sensor-rich Videos in Geo-space. In *3rd Multimedia Systems Conference*, pages 53–64, 2012. 4