Generating Synthetic Meta-data for Georeferenced Video Management

Sakire Arslan Ay
Department of Computer Science
University of Southern California
Los Angeles, CA 90089
arslan@usc.edu

Seon Ho Kim
Integrated Media Systems Center
University of Southern California
Los Angeles, CA 90089
seonkim@usc.edu

Roger Zimmermann
School of Computing
National University of Singapore
Singapore 117417
rogerz@comp.nus.edu.sg

ABSTRACT
Recently various sensors, such as GPS and compass devices, can be cost-effectively manufactured and this allows their deployment in conjunction with mobile video cameras. Hence, recorded clips can automatically be annotated with geospatial information and the resulting georeferenced videos may be used in various Geographic Information System (GIS) applications. However, the research community is lacking large-scale and realistic test datasets of such sensor-fused information to evaluate their techniques since collecting real-world test data requires considerable time and effort. To fill this void, we propose an approach for generating synthetic video meta-data with realistic geospatial properties for mobile video management research. We highlight the essential aspects of the georeferenced video meta-data and present an approach to simulate the behavioral patterns of mobile cameras in the synthetic data. The data generation process can be customized through user parameters for a variety of GIS applications that use mobile videos. We demonstrate the feasibility and applicability of the proposed approach by providing comparisons with real-world data.

Categories and Subject Descriptors
H.2.8 [Database Management]: Database Application—Spatial databases and GIS; H.3.4 [Information Storage and Retrieval]: Systems and Software—Performance evaluation (efficiency and effectiveness)

General Terms
Algorithms, Experimentation, Performance

Keywords
GIS, georeferenced video, meta-data generation

1. INTRODUCTION
Technological trends have enabled the cost- and energy-efficient deployment of video cameras together with other sensors (e.g., GPS and compass units) on mobile platforms such as handheld devices or vehicles of various forms. As a result, videos have gained mobility and attained geospatial properties through the association with sensor meta-data. This raises many interesting research challenges for the Geographic Information Systems (GIS) community. As an example, recently smartphones have become extremely popular in capturing and sharing videos. Because of the affordability of these camera-equipped gadgets, the general public is now generating and disseminating an extensive amount of video clips.

The scenes captured in a video change with the movement and rotation of the camera. Figure 1 exemplifies the scene coverage of a video on a map, based on the associated GPS and compass sensor values. These geographically described (i.e., georeferenced) video data contain significant information about the region where they were captured and can be effectively utilized in various GIS applications to, for example, visually navigate and query the geo-space [11]. Potential mobile video applications are not limited to smartphones. Recently, some service vehicles such as city buses, ambulances, and police cars have been equipped with cameras for various reasons, e.g., to monitor the quality of service provided. These cameras record a vast amount of data that cover large geographic areas. It is often desirable to browse, summarize and search these videos to find the video sections that capture a particular region or a certain location.

Figure 1: Map visualization of video scene coverage.
at a semantically interesting level and enable effective and efficient management of mobile videos. For example, there have been studies on organizing and browsing images and videos according to their locations [15, 12, 5]. Some of these techniques are evaluated using the georeferenced images from popular image sharing services such as Flickr [12]. These community-contributed datasets are mostly based on user-provided geo-tags, therefore they may be subjective and ambiguous. Furthermore, each image/video is associated with only a single geographical location, which may not be sufficient for mobile videos. In our earlier work [1, 2] we proposed the use of geographical properties of video scenes as an effective means to aid in the search of large video archives. Subsequently, we introduced a camera prototype to capture videos along with location and direction meta-data, automatically annotated from sensors [1]. We collected many hours of georeferenced videos using our recording system for the evaluation of our mobile video management techniques. Other studies have generated their own georeferenced meta-data [10, 9, 5]. Such real-world datasets are essential for the evaluation and comparison of various mobile video management techniques. However, the limitation of these datasets is that they were all generated within a controlled environment, therefore the behavior of the video capture can be perceived as subjective. Moreover, due to a highly labor intensive collection process, the amount of georeferenced real-world video data available has not been large enough to evaluate realistic application scenarios. To enable comprehensive performance evaluations on a large scale, much bigger datasets are required. Consequently, one important and urgent requirement to facilitate future research activities is the availability of well-described georeferenced video meta-data, enabling a systematic and extensive comparison of mobile video management techniques. However, collecting real-world data requires a considerable amount of time and effort. While there is a need for a collaborative effort to build a large repository of georeferenced videos, a complementary solution to fulfill the urgent requirement of the research community is to synthetically generate georeferenced video meta-data.

In this paper, we propose an approach for generating synthetic video meta-data with realistic geographical properties for mobile video management research. We provide our observations about the behavior of mobile cameras and how to emulate such behavior when generating synthetic data. Note that the generated meta-data is based on the definition of our viewable scene model [1] that is comprised of a camera’s position in conjunction with its viewing direction, viewable distance and zoom level. Users can control the behavior of the proposed generator using various input parameters. Our contributions in this work include:

- Identification and classification of the requirements on generating synthetic georeferenced video meta-data.
- Design of algorithms to synthetically generate practical video meta-data with requested geospatial properties. In particular, the generation of video meta-data with realistic camera movements and camera rotations.
- Customization of the generation process based on user parameters for broader applications.
- Comparison and evaluation of the characteristics of the real-world and synthetic video meta-data.

The rest of the paper is organized as follows. Section 2 contains a brief summary of the main concepts related to georeferenced mobile video management. The requirements for generating synthetic video meta-data are discussed in Section 3. This is followed by the detailed description of the synthetic data generation process in Section 4. Section 5 presents the results of the comparison of the real-world and synthetic datasets. Section 6 provides a survey of related work. Finally, Section 7 concludes the study and discusses open issues.

2. BACKGROUND

2.1 Georeferenced Video Annotations

We define the scene that a camera captures as its viewable scene. In the field of computer graphics, this area is referred to as camera field-of-view (FOV). We describe a camera’s viewable scene in 2D space using the following four parameters: 1) the camera position P consisting of the (latitude, longitude) coordinate read from a positioning device (e.g., GPS), 2) the camera direction vector \( \vec{d} \) which is obtained from the orientation angle provided by a digital compass, 3) the camera viewable angle \( \theta \) which describes the angular extent of the scene imaged by the camera [7], and 4) the visibility range \( R \) which is the maximum distance from the camera at which a large object within the camera’s field-of-view can be recognized. Then the camera viewable scene at time \( t \) is denoted by the tuple \( FOVScene(t, P, \vec{d}, \theta, R) \). In 2D space, the viewable scene of the camera forms a pie-shaped area as illustrated in Figure 2a. Figure 2b shows an example camera \( FOVScene \) volume in 3D space. For a 3D representation of \( FOVScene \), we would need the altitude of the camera location point (from the GPS device) and the pitch and roll values (from the compass) to describe the camera heading on the \( xx \) and \( yy \) planes (i.e., whether the camera is directed upwards or downwards). In this study we will represent the \( FOVScene \) in 2D space only. However, the data generator could be extended to 3D.

![Figure 2: Illustration of FOVScene model (a) in 2D and (b) in 3D.](image_url)
According to the required level of details in the videos, an appropriate $R$ value can be used.

In large scale applications, there may be thousands of moving cameras with different sensing capabilities. Each camera will record its FOVScene with a different sampling frequency. Intuitively, there should be a separate FOVScene quintuple for every video frame. According to the required granularity of the video meta-data, an appropriate sampling frequency can be determined by the user. Note that a higher sampling rate creates more detailed video scene descriptions, but requires a larger storage space. The assigned sampling rate should balance this tradeoff. Further details on timing and sampling frequency of the meta-data can be found in our earlier work [1].

### 2.2 Real-world Video Collection

To collect georeferenced video data, we have constructed a prototype system which includes a camera, a 3D compass and a GPS receiver. We used the JVC JY-HD10U camera which produces MPEG-2 HD video streams available in real time from the built-in FireWire (IEEE 1394) port with 30 frames per second data rate. To obtain the orientation of the camera, we employed the OS5000-US Solid State Tilt Compensated 3 Axis Digital Compass, which provides precise tilt compensated headings with roll and pitch data. To acquire the camera location, the Pharos iGPS-500 GPS receiver has been used. A program was developed to acquire, process, and record the georeferences along with the MPEG-2 HD video streams. The system can process MPEG-2 video in real-time (without decoding the stream) and each video frame can be associated with its viewable scene information. More details on synchronization of meta-data with video content have been provided in our earlier work [1]. Figure 3a shows the setup for our recording prototype. We have mounted the recording system setup on a pickup truck and captured video outdoors in Moscow, Idaho, travelling at different speeds. For the real-world dataset ($RW$) the average camera movement speed was around 27 km/h ($\approx 17$ mph), and the average camera rotation was around 12 degrees/sec. One meta-data tuple was generated every second, i.e., one FOVScene($t$, $P$, $d$, $\theta$, $R$) per 30 frames of video. We collected a total of 170 minutes ($\approx$ 3 hours) of video data that covered a 6 km by 5 km region quite uniformly. The dataset contained 10,652 FOVScene tuples and its total size was approximately 23 MB. In order to evaluate the performance of several research techniques regarding georeferenced videos, much larger datasets are needed. However, collecting real-world data requires considerable time and effort, even for small datasets.

We also implemented a prototype georeferenced video acquisition module on a smartphone, an Apple iPhone 3GS handset, which provides the necessary built-in GPS receiver and compass functionality (see Figure 3b). We plan to collect more real-world data with this platform for further studies in the future.

### 3. Synthetic Video Meta-data Generation Requirements

We next review the requirements for the generation of georeferenced video meta-data. We specifically investigate the behavior of mobile cameras and how to simulate such behaviors in the synthetic data. Our goal is not to provide a case study generator that fulfills the requirements of a certain specific application. Instead, we present a customizable system which the user can configure according to the required meta-data properties for more general applications. The highlighted requirements are based on our analysis on the collected real-world data.

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Trajectory →</td>
<td>Type of the trajectory that the camera follows pattern.</td>
</tr>
<tr>
<td>Rotation →</td>
<td>Property that describes the behavior of the camera rotation.</td>
</tr>
<tr>
<td>Visual angle →</td>
<td>The visual angle for the camera field-of-view.</td>
</tr>
<tr>
<td>Visibility →</td>
<td>The visible distance at which a large object range within the camera’s field-of-view can still be recognized.</td>
</tr>
</tbody>
</table>

Table 1: Summary of camera template specification.

In an actual video capture, the behavior of a camera (i.e., its movements and rotations) depends on the occasion and purpose of the video recording. Such camera behavior can be described by the pattern of movements and rotations, which we refer to as a camera template in this paper. A camera template overview is given in Table 1. In the template definition, the trajectory pattern property identifies the movement pattern of the camera trajectory. Example patterns include, network-based when a camera moves along a predefined network such as roads, free movement when a camera has no restriction in moving directions, or a combination of the two. The rotational behavior is described by the rotation pattern property. If the camera rotation is not allowed, this property is set to fixed and if the camera can be freely rotated (e.g., by a human user), this property is set to random. The visual angle and visibility range correspond to the parameters $\theta$ and $R$ in the scene model, respectively. For a particular camera, we assume that all the georeferenced meta-data streams will have identical $\theta$ and $R$ values. Consequently, in generating synthetic meta-data, the generator produces meta-data only for the camera trajectory and rotation. Table 2 provides the detailed list of the camera template parameters, their grouping and also their corresponding notations.

Below we provide three typical example templates for mobile videos:

1. **Vehicle camera**. The camera is mounted on a vehicle and moves along road networks. Examples include cameras equipped on city buses, police cars or ambulances. The trajectory pattern is network-based. The movement speed is assumed mostly fast and steady.
The camera heading with respect to the direction of vehicle movement is fixed (i.e., the rotation pattern is fixed). Thus, the camera direction changes only when the vehicle makes turns.

2. **Passenger camera.** A passenger traveling in a vehicle operates the camera. Examples include hand-held cameras used to capture scenery, buildings, landmarks, etc. The camera trajectory pattern is set to network-based. The camera bearing changes when the moving direction changes and when the user rotates the camera, i.e., the rotation pattern is specified as random.

3. **Pedestrian camera.** A walking pedestrian holds the camera. Examples include hand-held cameras used to capture tourist attractions, landmarks, and special events. The camera mostly follows a random trajectory, analogous to a walking path. However, it might also follow a predefined network, such as sidewalks or hiking trails. The trajectory pattern is a combination of network-based and free movement. The camera can be rotated freely, therefore the rotation pattern is set to random.

---

### Table 2: Details of camera template parameters.

#### 3.1 Camera Movement Requirements

For the camera templates described above, there are three major behavioral patterns (i.e., trajectory patterns) for the camera movement:

1. $T_{\text{network}}$: Moving on a given road network.
2. $T_{\text{free}}$: Moving on an unrestricted path.
3. $T_{\text{mixed}}$: Partly following a network and partly moving on a random path.

In order to realistically simulate the movements, some other practical properties are required in addition to the trajectory patterns, such as: 1) maximum speed ($V_{m}^{\text{road}}$, $V_{m}^{\text{walk}}$), 2) frequency and duration of stops ($P_{e}$, $t_{s}$), and 3) the camera acceleration and deceleration parameters ($n$, $p$, $A$), etc.

Table 2 lists the required camera template parameters to configure the camera movements of the trajectory types.

A mobile camera is an example of a moving object whose positions are samples at points in time. In the field of spatio-temporal databases, there exists an extensive collection of research for generating synthetic moving object test data for evaluating spatio-temporal access methods. We adopt and revise some well-known moving object data generation techniques for the proposed synthetic data generation. Details on generation of camera movements will be described in Section 4.1.

#### 3.2 Camera Rotation Requirements

There can be certain restrictions for a practical simulation of rotational behaviors, even for freely rotating camera directions. These restrictions are required to be enforced by assigning appropriate values to certain camera template parameters (see Table 2). For example, $\text{offset}$ parameter defines the initial bearing of the camera. If the camera direction is fixed then the camera direction will be analogous to the moving direction, maintaining the angle $\text{offset}$ with respect to the moving direction. Often, to ensure visual quality in captured videos, the rotation speed is limited within a certain threshold. $\omega_{\text{max}}$ parameter defines the maximum allowable rotation per time unit. For the random rotation pattern, the user might want to restrict the maximum rotation with respect to the offset. The $\alpha_{\text{max}}$ parameter defines the maximum allowed rotation around the offset.

In collecting real world georeferenced meta-data, the camera location and direction values are acquired from two independent sensors. However, based on our analysis on the real data, we observed that the movement and rotation of a camera is correlated. For example, cameras mostly point towards the movement direction (with some varying offset). In addition, when the camera trajectory makes a turn at a corner or a curve, the camera direction also changes. For fast moving cameras (e.g., a camera mounted on a car) there is a close correlation between the movement speed and the amount of rotation. Such relationships are caused by human behavior in controlling the camera rotation. The faster the camera moves the less it rotates, mainly because human operators try to avoid fast horizontal panning such as to cautiously maintain the quality of videos. Figure 4 illustrates this observation based on the real world data. The average rotation of a camera linearly decreases as its movement speed increases.

![Figure 4: Analysis of real-world data: average rotation as a function of the camera speed.](image-url)
The aforementioned properties surely cannot consider all possible aspects of mobile video recording. The characteristics of georeferenced video data may widely vary depending on when, where, by whom and for what purpose the video is captured. Users may want to create and customize new camera templates by defining different movement and rotation specifications. Hence, a synthetic data generator needs to be able to fulfill these requirements. By giving users the flexibility to customize the camera behavior, our data generator can provide synthetic data applicable to a wide range of application scenarios.

4. VIDEO META-DATA GENERATION

Our methodology for generating synthetic georeferenced meta-data is based on the observations stated in Section 3. We generate the synthetic data in two major steps (see Figure 5): First we create camera movement trajectories, and then in the second step we estimate the camera direction for each location on the movement path.

4.1 Generating Camera Movement

To create the camera trajectories, we adopted two widely accepted spatio-temporal data generators for moving objects: the Brinkhoff generator [3] and the GSTD generator [13]. The former is used to generate the trajectories of a camera that moves on a road network (T\textsubscript{network}), while the later is used to simulate the trajectories of a pedestrian (T\textsubscript{free}). The basic behaviors of the generators are controlled by a set of parameters, however, more sophisticated additional changes for the georeferenced video management research required additional considerations. In order to ensure that the generated camera trajectories from the Brinkhoff and GSTD algorithms had the required specifications, we modified both software algorithms and added some features.

4.1.1 Network-based Camera Movement (T\textsubscript{network})

We generate network-based trajectories using the Brinkhoff algorithm with the required maximum speed \(V_m^{road} \)\, given a camera template. The array \(V_m^{road} \) defines the maximum allowed speeds for traveling on freeways, main roads and side roads. The Brinkhoff generator adjusts the speed of a moving object based on the maximum speed allowed on the current edge (i.e., road) as well as the load on that edge (the current number of objects using it). The Brinkhoff generator uses TIGER/Line files \[17\] to build the road network. The routes that the moving objects follow are determined by selecting the best network edges between the starting and destination points. The objects always try to move at the maximum possible speed and their movement is continuous unless there is congestion on the current edge. The Brinkhoff algorithm does not introduce any stops at intersections (e.g., at traffic lights and stop signs). In addition, there is no support for modeling the acceleration and deceleration of the objects. Thus, we modified the original algorithm and inserted stops and decelerations at some road crossings and transitions. The probability of having a stop event at an intersection is given by \(P_s\). In a deceleration event, at every time unit the cameras’ current speed \(v\) is reduced to \(v \times B(n, p)/n\) where \(B(n, p)\) is a binomially distributed random variable \((n\) is the number of experiments and \(p\) is the probability for a positive outcome). In a typical run, we use \(n = 20\) and \(p = 0.5\), which results in a 50% speed reduction at each time unit \[4\]. In a stop event, the camera decelerates and reduces its speed to 0 km/h. After a full stop, it stays steady for a duration of \(t_s\) seconds. The acceleration event occurs when the available speed limit on the current edge increases or when the camera starts to move after a stop event. The cameras accelerate with a constant acceleration rate of \(A m/s^2\) \[4\]. All \(V_m^{road} \), \(P_s\), \(t_s\), \(n\), \(p\), and \(A\) values are obtained from the camera template definition. The output for the \(T_{\text{network}}\) trajectories includes the timestamped camera location updates with a sampling rate of \(f_{\text{sample}}\). It generates \(N\) trajectories with a maximum trajectory length of \(L\). \(N\) and \(L\) values are input parameters to the generator.

4.1.2 Unconstrained Camera Movement (T\textsubscript{free})

We use the GSTD algorithm to generate the camera trajectories with unconstrained movement. GSTD was one of the first generators for creating moving object data. The behavior of movements can be customized by adjusting the domain of displacements \(\Delta x\) and \(\Delta y\), i.e., the minimum and maximum possible movements on the x and y axes. The object movement can either be random or partly restricted on one axis (i.e., the objects are forced to move towards a particular direction). To ensure a certain speed level for the camera trajectories, we modified the algorithm and added a speed control mechanism. The user provides the desired maximum speed \(V_m^{walk} \) in km/h in the camera template specification. Then, the generator calculates the object speed for every location point on the trajectory and ensures that overall, the user speed requirements are fulfilled. To simulate the movement of a pedestrian, the maximum walking speed is set to 5.43 km/h. The initial locations of the cameras are calculated based on the distribution function \(U\), which is given in the camera template. It is important to note that GSTD represents object locations as relative distances with respect to the mid-point of the region. However, our georeferenced video model \[1\] requires the location meta-data in the geographic coordinate system (i.e., latitude/longitude coordinates). Therefore, we also modified the trajectory representation of GSTD and generated object movements with latitude/longitude coordinates. Such a representation enables the monitoring of the object speed given that the speed requirements are defined in common metric units (i.e., km/h). In addition, unlike the original GSTD algorithm, we report location updates in the trajectory periodically with a parameterized sampling rate \(f_{\text{sample}}\).

4.1.3 Mixed Camera Movement (T\textsubscript{mixed})

The trajectories in this category are combinations of sub-trajectories from \(T_{\text{network}}\) and \(T_{\text{free}}\). Cameras sometimes
follow the network and sometimes move randomly on an unconstrained path. Algorithm 1 formalizes the trajectory generation of $T_{mixed}$. We first generate a $T_{network}$ trajectory (line 2), then we randomly select $n$ point pairs $S = \{(p_{i1}, p_{i2}) | 1 \leq i \leq n\}$ on the trajectory, where each point pair $S_i$ represents a segment on the trajectory. The duration of each segment $\{(S_i)\}$ can change between 0 and $|T_{init}|/4$, where $|T_{init}|$ is the length of the original $T_{network}$ trajectory. Function GetLineSegments() randomly selects $n$ subsegments in $T_{init}$, such that $N_{rand} = (\sum_{i=1}^{n} S_i)/|T_{init}|$. Recall that, $N_{rand}$ is the camera template argument that specifies the percentage of random movement in the generated $T_{mixed}$. The distribution of the segments on the trajectory is given by the camera template argument $D_{rand}$. It can be either uniform, Gaussian (around the middle), or a mixture of Gaussian. The function CreateRandTraj() implements a restricted version of GSTD, where the starting and end points are predefined. Note that if $p_{i1} = p_{i2}$, nothing is removed. One important issue in creating a $T_{mixed}$ trajectory was merging the trajectories from $T_{network}$ and $T_{free}$ together. We made sure that the trajectories from both groups have the same output format. Also, it is not necessarily true that the removed and inserted segments are of equal time length (i.e., $|S(i)| = |T_{rand}(i)|$). Therefore, there might be overlaps and/or gaps in the timestamps of the combined trajectories. Algorithm 1 revises the timestamps in $T_{mixed}$ (function AdjustTimestamps()) and makes sure that the timing within the final trajectory is consistent.

Algorithm 1 MixedMovement($netT_{arg}$, $randT_{arg}$, $CAM$)
1 $CAM$ ← camera template specification, $netT_{arg}$ ← arguments for creating $T_{network}$, $randT_{arg}$ ← arguments for creating $T_{free}$.
   Returns: $T_{mixed}$ trajectory.
2 $T_{init} = \text{CreateNetTraj}(netT_{arg}, CAM)$
3 $|S| = \text{GetLineSegments}(T_{init}, CAM)$
4 for all $S(i) = (p_{i1}, p_{i2}) | 1 \leq i \leq n$ do
5   $T_{rand}(i) = \text{CreateRandTraj}(randT_{arg}, p_{i1}, p_{i2})$
6   $T_{free} = T_{init}.\text{Replace}(p_{i1}, p_{i2}, T_{rand}(i))$
7 end for
8 return AdjustTimestamps($T_{mixed}$);

4.2 Generating Camera Rotation

An important difference between the traditional moving object data generators and our work is the computation of the camera direction. Assigning meaningful camera direction angles for the location points along the trajectory is one of the novel features of the proposed data generator. The assignment of direction angles is customized following the specifications provided by users (see Table 2).

4.2.1 Generating Camera Direction Angles

As shown in Figure 5, after the generator creates the camera trajectories, it assigns appropriate direction angles for all sampled points on the trajectories. Algorithm 2 formalizes the assignment of camera directions. The camera movement trajectory $T_k$ and camera template specification $CAM$ are given to the algorithm as input. First, the algorithm assigns the camera direction to be the moving direction and stores the camera trajectory with moving directions in array $M_k$. Next, it adjusts the rotations that exceed the $CAM.\omega_{max}$ threshold (lines 3-6). If the camera can be freely rotated (i.e., random rotation), the algorithm assigns an appropriate random direction angle for each point in $M_k$ (function RandomizeDirections()). The amount of rotation is computed with regard to the current camera speed and the specifications defined in $CAM$.

Algorithm 2 $M_k = \text{AssignDirections}(T_k, CAM)$
1 $T_k$ ← The trajectory array for $V_k$.
2 $CAM$ ← Camera template specification.
   Returns: $M_k$ ← The meta-data array for $V_k$.
3 Copy $T_k$ to $M_k$;
   {Initially set the direction for each data point to “moving direction + $CAM.\omega_{max}$”}
4 for $i$ = 1 to $|T_k|$ do
5   $T_k(i).\text{dir} = \text{GetMovingDirection}(T_k, i, CAM);$
6 end for:
7 AdjustRotations($M_k, CAM$);
8 if $CAM.\text{rotation_pattern}$ is “random” then
9   RandomizeDirections($M_k, CAM$);
10 end if
11 return $M_k$;

4.2.2 Calculating Moving Direction

The moving direction of the camera is estimated as the direction vector between two consecutive trajectory points. Given a camera trajectory $T_k$, the moving direction at sample point $t$ is given by the unit vector, $\vec{m} = T_k(t-1)-T_k(t)$. The camera direction angle (with respect to North) at sample point $t$ can be obtained by:

$$T_k(t).\text{dir} = \text{cos}^{-1}\left(\frac{\vec{m}.y}{|\vec{m}|}\right) + \text{offset}$$

In Algorithm 2, the function GetMovingDirection() calculates the moving direction angle at all trajectory points and initializes the direction of the moving direction plus the given offset ($CAM.\omega_{offset}$). Next, the function AdjustRotations() checks whether the rotations due to the changes in the moving direction are below the required threshold $CAM.\omega_{max}$. The synthetically generated camera trajectories often have sharp turns which results in abrupt rotations in the camera moving direction, which is unusual in reality. Our observation on real-world data shows that the change in the moving direction is much smoother. As an example, for vehicle and passenger cameras, when the vehicles turn at corners and steep curves on the road network, the moving direction will gradually change and it will take several seconds to complete the turn. Figure 6a shows an example trajectory from the real data where the car makes a left turn. The change in the speed is illustrated with different color codes, i.e., red points mark the slowest and blue points the fastest speeds. The arrows show the moving directions. Figure 6b illustrates a trajectory synthetically generated for the same road intersection. When the point moving direction is estimated from the movement vector, the rotation angle within a single time unit can be as large as the right angle (the rotation at the corner point $\geq 90^\circ$ in Figure 6b). In order to simulate the direction change during a turn, the generator distributes the total rotation amount among several trajectory points before and after the corner point. As seen in Figure 6c, with this method the moving direction changes gradually, and the rotation between two consecutive points is guaranteed to be under $CAM.\omega_{max}$. The camera locations are also updated based on the adjusted moving directions.
The function AdjustRotations() in Algorithm 2 checks for the rotation angle at each sample point, and smooths the sharp turns by interpolating the rotation angle among several neighboring sample points. Algorithm 3 formalizes the adjustment of rotations. The meta-data array $M_k$ with moving directions is given to the algorithm as input. The algorithm checks the rotation amounts between all consecutive trajectory points in $M_k$. If the rotation amount exceeds $\text{CAM}.\omega_{\text{max}}$ at point $M_k(i)$, then the algorithm scans the trajectory backwards and forwards until it finds two trajectory points $M_k(i-\Delta_{\text{bwd}})$ and $M_k(i+\Delta_{\text{fwd}})$ such that the camera can be safely rotated from $M_k(i-\Delta_{\text{bwd}})$ to $M_k(i+\Delta_{\text{fwd}})$ without violating the $\text{CAM}$ requirements. When such points are found the directions between points $M_k(i-\Delta_{\text{bwd}})$ and $M_k(i+\Delta_{\text{fwd}})$ are interpolated (line 18). Given a sample point $j$, function InterpolateAngle() returns the $j^{th}$ interpolated angle, i.e.,

$$M_k(i-\Delta_{\text{bwd}}).\text{dir} + j \left( \frac{M_k(i+\Delta_{\text{fwd}}).\text{dir} - M_k(i-\Delta_{\text{bwd}}).\text{dir}}{\Delta_{\text{fwd}}+\Delta_{\text{bwd}}} \right)$$

It guarantees that the returned direction is between $\text{CAM}.\omega_{\text{max}}$ and $\text{CAM}.\omega_{\text{max}}$.

**Algorithm 3 AdjustRotations($M_k, \text{CAM}$)**

1. $M_k$ — The meta-data array for $V_k$,
2. $\text{CAM}$ — Camera template specification.
3. Returns: $M_k$ with revised directions.
4. for $i = 2$ to $|M_k|$ do
5.   if AngleDiff($M_k(i-1).\text{dir}$, $M_k(i).\text{dir}$) > $\text{CAM}.\omega_{\text{max}}$ then
6.     $\Delta_{\text{fwd}} = 1; \Delta_{\text{bwd}} = 0$;
7.     {Scan backwards and forwards to find two data points such that the rotation between them is possible without exceeding $\text{CAM}.\omega_{\text{max}}$}
8.     repeat
9.       $\text{dir change} = \text{AngleDiff}(M_k(i+\Delta_{\text{fwd}}).\text{dir}, M_k(i-\Delta_{\text{bwd}}).\text{dir})$
10.      if $\Delta_{\text{fwd}} > \Delta_{\text{bwd}}$ then
11.         increase($\Delta_{\text{bwd}}$);
12.     else
13.         increase($\Delta_{\text{fwd}}$);
14.     until ($\text{dir change} < \omega_{\text{max}} * (\Delta_{\text{fwd}}+\Delta_{\text{bwd}})$);
15.      {Adjust all directions between ($i-\Delta_{\text{bwd}}$) and ($i+\Delta_{\text{fwd}}$) such that rotation amount do not exceed $\text{CAM}.\omega_{\text{max}}$ and $\text{CAM}.\omega_{\text{max}}$}
16.      for $j = 1;\Delta_{\text{bwd}}$ to ($i+\Delta_{\text{fwd}}$) do
17.        $M_k(i).\text{dir} = \text{InterpolateAngle}(j, M_k(i-\Delta_{\text{bwd}}).\text{dir}, M_k(i+\Delta_{\text{fwd}}).\text{dir}, \text{CAM}.\omega_{\text{max}}, \text{CAM}.\omega_{\text{max}})$
18.      end for
19.  end if
20. end for
21. return $M_k$.

4.2.3 Assigning Random Direction Angles

In Algorithm 2, if the camera’s bearing with respect to its moving direction is fixed, the function returns the meta-data array with adjusted moving directions. However, if the camera rotation is allowed, then the algorithm randomly rotates the directions at each sample point towards left or right. Given the initial camera directions based on the moving direction, function RandomizeDirections() updates direction angles to simulate the rotation of the camera by a user. Algorithm 4 details the randomization of the directions. It generates a random rotation amount which is inversely proportional to the current speed level (line 4). The faster a camera moves the less rotation there will be. In addition, the algorithm makes sure that the rotation amount is less than $\text{CAM}.\omega_{\text{max}}$ and the assigned camera direction angles are between $\text{CAM}.\omega_{\text{max}}$ and $\text{CAM}.\omega_{\text{max}}$ (line 5). The algorithm returns the meta-data array with updated directions.

**Algorithm 4 RandomizeDirections($M_k, \text{CAM}$)**

1. $M_k$ — The meta-data array for $V_k$,
2. $\text{CAM}$ — Camera template specification provided by user.
3. Returns: $M_k$ with revised directions.
4. for $i = 2$ to $|M_k|$ do
5.   repeat
6.     rotate = $\text{(getRandNum() \times getRandSign()) / M_k(i).\text{speed}}$;
7.     until ($|M_k(i).\text{dir} + \text{rotate} \leq \text{CAM}.\omega_{\text{max}}$) and
8.        ($|\text{CAM}.\omega_{\text{max}} - M_k(i).\text{dir} - \text{rotate} \leq \text{CAM}.\omega_{\text{max}}$)
9.     $M_k(i).\text{dir} = M_k(i).\text{dir} + \text{rotate}$;
10.   return $M_k$;
11. end for

4.3 Creating Meta-data Output

Our generator can either create the georeferenced meta-data output in text format or directly insert the meta-data tuples into a MySQL database. The output is the list of the sampled camera location ($P$) and direction ($d$) updates. Following the input sampling rate ($f_{\text{sample}}$), each meta-data item is assigned a timestamp. For timestamps we use the Coordinated Universal Time (UTC) standard format. The first data point in a meta-data stream is assigned the current UTC time. The rest of the timestamps are adjusted according to the given granularity in time based on $f_{\text{sample}}$.

5. EXPERIMENTAL EVALUATION

5.1 Comparison with Real-world Dataset

In this section we empirically examine some high-level properties of the synthetic georeferenced video meta-data. We analyze the movements and rotations of both the real-world and synthetic datasets. The effectiveness of the synthetic data generation approach is evaluated using a high-level comparison between the real-world and synthetic data obtained for the same geographical region.
5.1.1 Datasets and Evaluation Methodology

Using the proposed georeferenced video meta-data generator, we produced two groups of synthetic data, namely $S_{car}$ using the vehicle camera template and $S_{pass}$ using the passenger camera template. For each synthetic data group, we generated 10 datasets. For the first group ($S_{car}$), the cameras cannot be rotated, therefore the camera orientation is the same as the moving direction. For the second group ($S_{pass}$) the camera can be rotated freely. Both synthetic data groups were created based on the road network of Moscow, Idaho, where the real-world data were collected. The maximum speed and object class parameters for both $S_{car}$ and $S_{pass}$ were adjusted to make the maximum camera speed comparable to the real-world data. The average length of the video files (i.e., $L$) in the synthetic datasets was also similar to that of the real-world video files. The output sampling rate $f_{sample}$ was set to 1 sample/s. Table 3 summarizes the properties of the generated data. The values from the 10 datasets in each of the groups $S_{car}$ and $S_{pass}$ were averaged and we only report these averaged values in Table 3.

To evaluate the applicability of the synthetic datasets, we are interested to analyze the important characteristics of synthetic datasets and compare them to those of the real-world dataset ($RW$). We compare $S_{car}$ and $S_{pass}$ with $RW$ in terms of, 1) the speed of the cameras and 2) the rotation of the cameras. We report the average and maximum values for speed and rotation, and the frequency distribution of the different speed and rotation levels within the datasets. Note that the properties of the real-world dataset is highly subjective and depends on how the user moves and rotates the camera. During real-world data collection, we rotated the camera as needed to capture important scenes and avoided fast movements and rotations.

5.1.2 Comparison of Camera Movement Speed

First, we analyze how fast and how rapidly cameras move. When we generated $S_{car}$ and $S_{pass}$, we set the maximum allowable speed parameter $V_{max}^{road}$ [1] based on the speed limits of the different types of roads in the region (i.e., 88 km/h for highways, 56 km/h for main roads, and 40 km/h for side roads). Recall that the generator calculates the maximum allowable speed on different edges of the road network based on this parameter. The other camera template parameter values are $A = 12$ m/s² [4], $P_c = 0.3$, and $t_s = 3s$. For all 10 datasets in $S_{car}$ and $S_{pass}$, we calculated the absolute speed of the camera at each trajectory point. Table 4 reports the maximum and average values and the standard deviation of the average speed. The values are averaged for the 10 datasets in $S_{car}$ and $S_{pass}$. We observe that the average and standard deviation values for both $S_{car}$ and $S_{pass}$ are very close to the values of $RW$. This implies that the point speed distributions of the synthetic and real-world datasets are quite similar. To further investigate this, we constructed the histogram for different speed ranges in $S_{car}$ and $RW$. Figure 7 shows the percentages of the trajectory points for various speed ranges. While $S_{car}$ and $RW$ have similar trends, for $S_{pass}$, the majority of the trajectory points have speed values around the average speed, whereas for $RW$ the distribution of the camera point speed is more uniform. In dataset $S_{car}$, cameras always try to travel at the maximum allowed speed. They only slow down or stop when they are required to do so by traffic lights, stop signs, etc. The cameras always accelerate/decelerate with a certain rate. On the other hand, in $RW$ cameras usually speed-up and slow down with a lower acceleration/deceleration rate resulting in smoother speed transitions. This is due to the fact that human operators were very cautious in driving a car during the real-time video capture. Note that the trend in the speed histogram might change for different real-world datasets. The average speed limit for the road network used in the data generation was around 40 km/h ($\approx 25$ mph). This explains the dense population of the trajectory points between speed levels 20-40 km/h. Figure 8 illustrates the camera trajectories of the real-world dataset and a sample synthetic dataset. The color highlights on the trajectories show how fast cameras are moving on the road network. The blue color indicates the fastest speed while the red color marks the slowest speed. Note that overall, the movement behaviors of the cameras in both datasets are similar.

<table>
<thead>
<tr>
<th></th>
<th>Maximum speed (km/h)</th>
<th>Average speed (km/h)</th>
<th>StdDev of speed (km/h)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Synthetic Data with Fixed Camera ($S_{car}$)</td>
<td>87.91</td>
<td>27.14</td>
<td>12.82</td>
</tr>
<tr>
<td>Synthetic Data with Free Camera Rotation ($S_{pass}$)</td>
<td>87.28</td>
<td>27.32</td>
<td>13.01</td>
</tr>
<tr>
<td>Real-world Data ($RW$)</td>
<td>83.50</td>
<td>27.03</td>
<td>13.68</td>
</tr>
</tbody>
</table>

Table 4: Characteristics of the camera speed.

Figure 7: Comparison of camera speed distributions for $S_{car}$ and $RW$.

The camera movements of both the $S_{car}$ and $S_{pass}$ datasets are based on road networks, therefore their trajectory generation is the same. We obtained similar results for the comparison of $S_{pass}$ with $RW$ as we did for $S_{car}$ and $RW$. We omit these results due to space considerations.

5.1.3 Comparison of Camera Rotation

Next, our objective is to examine the properties of the camera rotations in the synthetic datasets and analyze the practicality of the data generator by comparing the rotation characteristics of $S_{car}$ and $S_{pass}$ with $RW$. We calculated the absolute rotation (in degrees/s) of the camera at each trajectory point for all 10 datasets in $S_{car}$ and $S_{pass}$. In Table 5, we report the maximum and average rotations as well as the standard deviation of rotations within each dataset. Again, the values were averaged for the 10 datasets in $S_{car}$ and $S_{pass}$. The maximum rotations obtained from $S_{car}$ and $S_{pass}$ were limited by the rotation thresholds provided to the data generator as inputs (i.e., 30 and 60 degrees/s, respectively). For $RW$, the maximum rotation was relatively large. Although not frequent, there were a few fast rotations in the
<table>
<thead>
<tr>
<th>Number of video files</th>
<th>Total number of trajectory points</th>
<th>Average video length (s)</th>
<th>Maximum speed (km/h)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Synthetic Data with Fixed Camera ($S_{car}$)</td>
<td>40</td>
<td>11849</td>
<td>299</td>
</tr>
<tr>
<td>Synthetic Data with Free Camera Rotation ($S_{pass}$)</td>
<td>40</td>
<td>11946</td>
<td>300</td>
</tr>
<tr>
<td>Real-world Data ($RW$)</td>
<td>25</td>
<td>10170</td>
<td>426</td>
</tr>
</tbody>
</table>

Table 3: Properties of the synthetic and real-world datasets.

![Illustration of camera movement speed on map](image)

Figure 8: Illustration of camera movement speed on map. (a) Real world data $RW$, (b) synthetic data $S_{car}$.

<table>
<thead>
<tr>
<th>Maximum rotation (degrees/s)</th>
<th>Average rotation (degrees/s)</th>
<th>StdDev of rotation (degrees/s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Synthetic Data with Fixed Camera ($S_{car}$)</td>
<td>32.33</td>
<td>4.64</td>
</tr>
<tr>
<td>Synthetic Data with Free Camera Rotation ($S_{pass}$)</td>
<td>55.27</td>
<td>12.59</td>
</tr>
<tr>
<td>Real-world Data ($RW$)</td>
<td>107.30</td>
<td>11.53</td>
</tr>
</tbody>
</table>

Table 5: Characteristics of the camera rotation.

real-world data resulting in larger values for maximum and standard deviation measurements. For $S_{car}$, since the cameras mostly travel on straight roads, the average rotation is quite small (<5 degrees/s). On the other hand for $S_{pass}$, where cameras can be rotated freely, the average rotation is very close to $RW$. Figures 9 and 10 show the percentage distributions of camera rotations for both real and synthetic data. In Figure 9, around 88% of the trajectory points rotate only 10 degrees or less per second. The remaining 12% correspond to the parts of the trajectories where the cameras make turns on the road network. As shown in Figure 10, the distribution of rotations for $S_{pass}$ is more uniform and the distribution trend is similar to $RW$. One important observation is that $RW$ exhibits higher percentages at lower rotation values, and lower percentages at higher rotations while percentages of trajectory points for $S_{pass}$ are more uniform among different rotation levels. This is due to the fact that in real-world data for some video sections the camera was rotated more frequently and more rapidly to capture certain attractions/landmarks. It was not rotated when there was no specific target to capture. As we mentioned earlier, such behavior is subjective. One possible improvement for our generator is to simulate different user behaviors for capturing the points of interest in the region.

5.2 Performance Issues

Here we provide a high-level analysis on the run time requirements of the generator. Our objective is to show that the generator can create synthetic datasets in a reasonable amount of time with off-the-shelf computational resources. Recall that the generator implements two major components: generating camera movements and assigning camera directions. For the trajectory generation, the performance of the proposed generator depends on the performance of the two underlying algorithms, Brinkhoff and GSTD, whose run time requirements have been examined before and are well-known. Therefore, we focus on the run time of the direction assignment algorithm (i.e., Algorithm 2).

We generated several synthetic datasets with different sizes and measured the computation time to produce each dataset. All measurements were performed on a 2.33 GHz Intel Core2 Duo Windows PC. Table 6 summarizes the measured times for different types of datasets and parameter settings. All data were generated for the same geographical region (i.e., Moscow, ID). In Table 6, the time for generating trajectories and assigning camera directions are listed separately. Overall the generator can produce very large datasets (containing millions of tuples) in a reasonable amount of time, i.e., within an order of minutes.

6. RELATED WORK

The main contribution of our work is on generating realistic synthetic meta-data to describe video viewable scenes. Conventional spatial database and GIS technologies can be used to partially generate some elements of video meta-data such as the location of a camera and the trajectory. Two main approaches have been proposed for generating moving object data. One of the earliest generators, GSTD [13], creates moving object trajectories with free movements. Many other simulators have been proposed to generate unconstrained object trajectories [14, 16]. The second approach assumes that the moving objects follow a predefined network. Brinkhoff [3] is a well known and widely used generator that produces moving object trajectories on a road network. Several other network-based trajectory generation techniques exist [6, 4]. New techniques are additionally required to generate camera direction information, which dynamically changes while the camera moves. Some techniques generate time-evolving 2D regional object data whose location, shape, size, and color change over time [16]. The changes in attributes are
based on constraints in the surrounding environment only. Conversely, in our work we also investigate the correlation between the location and direction of the viewable scene region. To the best of our knowledge, there has been no study to generate synthetic camera location and direction data to model geographical information of video scenes.

7. CONCLUSIONS AND FUTURE WORK

We presented an approach for generating synthetic video meta-data with realistic geospatial properties, based on the behavioral patterns of mobile cameras when they move and rotate. The requirements of the synthetic data generation were identified from the examples of real-world camera behaviors. Subsequently, we explained the data generation process and devised the algorithms used in meta-data creation. The high-level properties of the synthetically generated data and those of real-world georeferenced video data were compared. We concluded that the synthetic meta-data exhibit equivalent characteristics to the real data, and hence can be used in a variety of mobile video management research. As part of our future work we plan to investigate additional factors that might affect the camera behaviors and then to generate such behaviors in the synthetic data. An example can be the behavior of the camera when there are attractions, events or extraordinary occasions in the geo-space. The functionality of the proposed generator can be further extended by adding a user interface for map-based visualizations. We plan to make the generator software sources available to the research community.

Acknowledgments

We would like to acknowledge the support of the Centre of Social Media Innovations for Communities (CoSMIC), sponsored by the Media Development Authority (MDA) of Singapore.

8. REFERENCES


Table 6: Summary of the time requirements for synthetic data generation.

![Table 9: Comparison of camera rotation distributions for $S_{pass}$ and $RW$.](image1)

![Figure 9: Comparison of camera rotation distributions for $S_{car}$ and $RW$.](image2)

![Figure 10: Comparison of camera rotation distributions for $S_{pass}$ and $RW$.](image3)