ICASSP 2023 - 2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP) | 978-1-7281-6327-7/23/$31.00 ©2023 IEEE | DOI: 10.1109/ICASSP49357.2023.10094743

TRACKING TARGETS IN HYPER-SCALE CAMERAS USING MOVEMENT PREDICATION

Jiaping Yu[†] *Tongqing Zhou*[⋆†] *Zhiping Cai*[⋆†] *Wenyuan Kuang*[‡]

[†] College of Computer, National University of Defense Technology
[‡]360 Digital Security Group

ABSTRACT

Hyper-scale surveillance cameras have enabled seamless target (e.g., vehicles, individuals) tracking in urban scenarios, significantly improving everyday safety and emergency response capacity. However, practically tracking multiple targets among cameras would incur prohibitive huge computation costs, even with the acceleration utility of edge computing. This work observes with real-world camera deployment that existing tracking scheduling is unfortunately inefficient due to the redundant and over-activation of cameras. We then propose the design of a hierarchical tracking framework HyMOT that exploits fine-grained target movement prediction for efficient tracking in hyper-scale cameras. At its core, HyMOT builds a probabilistic target movement graph with tempo-spatial correlation knowledge extracted from historical statistics. With the graph, we formulate tracking scheduling as an optimization problem with efficiency-accuracy tradeoff constraints and solve this NP-hard problem with a greedy strategy. Experiments based on the Cityflow and Geolife datasets demonstrate that, compared with two baselines, HyMOT requires significantly less (over 90%) computation workloads to track the same amount of targets with similar accuracy.

Index Terms— Multi-Camera Multi-Target Tracking, Internet of Things, Sensor Scheduling.

1. INTRODUCTION

The deployment of hyper-scale surveillance cameras helps to make cities smarter and safer [1–3]. As reported, megacities like New York and Singapore have deployed thousands of surveillance devices [4]. Analyzing live videos streamed from these distributed cameras for object tracking is the backbone of safety-related applications [5–8]. For example, the BriefCam application [9] is designed for social distance tracking in the recent Covid-19 pandemic based on surveillance videos.

It is non-trivial to analyze videos from hyper-scale cameras. For example, the generated video of 100 distributed cameras with resolution 1080P can reach hundreds of gigabytes per minute [10]. Transmitting them to the remote cloud will cause tremendous pressure on the networks [11]. Hence, in practice, edge computing [12] is introduced to perform video analysis computation on network edge servers [13, 14]. So far, plenty of edge devices (e.g., Azure Stack Edge Pro R [15] have been equipped with DL computing hardware to support learning tasks on surveillance videos [16].

Even with onboard DL accelerators, tracking multiple targets on hyper-scale cameras still involves prohibitively high computation costs for edge servers, as we tested in Sec. 2. Therefore, attaining efficient target tracking has become a critical issue [17, 18]. Intuitively, selectively activating fewer edge servers and cameras for tracking a target could reduce the costs, while at the risk of sacrificing tracking accuracy. Recently, several tracking scheduling methods have been proposed by only activating cameras that are in the matched directions or with correlated neighborhood relationships with the moving targets [14, 19].

In this work, we observe with a real-world surveillance video dataset (CityFlow [10]) that existing scheduling strategies could still be cumbersome and provide limited efficiency when facing hyper-scale camera deployment. Three essential factors are elaborated for their inefficiency: 1) *Overlapping surveillance regions* of multiple cameras causes existing strategies' simultaneous activation redundant. 2) Cameras on the passageways are often *unnecessarily involved* when cameras on two ends of the passageway have already been invoked. 3) Cameras with *awkward locations and angles* are indiscriminately used, causing poor tracking accuracy and unavoidably more other cameras.

The above facts have never been investigated, to the best of our knowledge. To further improve the efficiency of signal analytics on sense visuals, we propose to exploit targets' movement predication with redundancy and correlation awareness and propose an efficient **Hy**per-scale **M**ulti-camera **O**bject **T**racking framework (HyMOT). It uses hierarchical geographic parsing that divides an edge server's administration region into multiple *Fields* of Interest, each representing an area covered by a cluster of nearby cameras, and divides a Field by multiple critical *Zones*, each denoting an entrance or exit area. With such a structure, HyMOT formulates the movement prediction of targets as a probabilistic movement graph that encodes tempo-spatial correlations as the number and frequency of inter-zone trips. Tracking costs are then quantified based on movement prediction and integrated with

⋆: Corresponding author

tracking accuracy into one tradeoff inequality. We take these probabilistic correlations as constraints and finally transform the efficient scheduling goal into linear programming of optimal edges and fields selection problem.

We have simulated both small-scale (7 Fields) and large-scale (100 Fields) camera networks based on real-world datasets Cityflow [10] and Geolife [20–22]. Experimental results on them show that, compared with Spatula and Anveshak, HyMOT could significantly reduce the computation burden (i.e., reducing at least 90% computation data) while maintaining intact tracking accuracy. This demonstrates the high feasibility of HyMOT in practical deployment.

2. ANALYSIS ON HYPER-SCALE TRACKING

Prohibitive costs. We first run a test to observe the computation costs of distributed edges on tracking objects. We simulate an edge server with a workstation equipped with an Intel i5 CPU, Nvidia GTX 960M GPU, and 4GB RAM. We extract the surveillance videos from the Cityflow dataset [10], which contains 1200-second of video from 6 cameras. It takes the simulated edge 4029 seconds to process with such an amount of data. For hyper-scale cameras for tracking, the incurred computation costs would be prohibitively high if directly activating all of them for real-time tracking on multiple targets.

Existing solutions. To reduce the computation in cross-camera object tracking, some researchers have proposed to carefully activate cameras in a target's nearby edge servers. Specifically, Anveshak [19] activates the surveillance cameras along the road that matches the moving direction and speed of the target. Likewise, Spatula [14] counts the targets passing through different surveillance regions and issues tracking tasks to the downstream camera that shares the most transferring objects with the current camera.

Observations on inefficiency. The above scheduling solutions still provide limited efficiency. By digging into the city-scale camera deployment in practice, we observe that the inefficiency can be owed to three aspects, as shown in Fig. 1. First, many cameras have *surveillance region overlapping*, making simultaneous activation of them for tracking redundant. Second, a camera between two cameras with no road detour cannot provide additional tracking tracklets, so is **unnecessary** to be activated. Third, some cameras have awkward **deployment locations and angles** which is useless in inferring targets' moving trajectories.

3. DESIGN OF HYMOT

We propose HyMOT, a new movement prediction-based tracking framework (Fig. 2) that addresses the inefficiencies of existing systems. HyMOT has a three-tier structure consisting of a scheduling edge (server), tracking edge (servers), and cameras. The scheduling edge generates the tracking schedule based on target movement prediction (Stage 1),

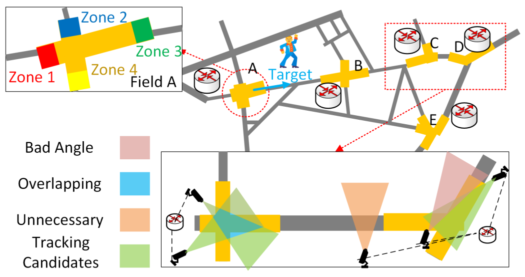


Fig. 1. A toy example on the tracking-inefficient deployment of surveillance cameras under distributed edge servers. Some camera activation is redundant, overused, and unnecessary in existing tracking solutions.

which is then sent to a group of tracking edges (Stage 2). The tracking edges assign target tracking tasks to local cameras based on appearance probabilities (Stage 3), and the camera reports target visuals when detected (Stage 4). The tracking edge that successfully detected the target in the previous round becomes the scheduling edge in the next round.

The scheduling strategy is at the core of HyMOT. Formally, it is denoted as the following minimization problem:

Hyper-scale Tracking Costs: Assuming that edge $S \in \mathbf{E}$ is the scheduling edge, responsible for tracking the $|\mathbf{J}|$ targets, and the tracking time period for the j-th target in the k-th edge is from t_{ak}^{j} to t_{bk}^{j} with $\Delta t_{k}^{j} = t_{bk}^{j} - t_{ak}^{j}$. Then we can calculate the amount of data needed to be processed in k-th edge when tracking target j as:

$$l_k^j = M \cdot D_k^j \cdot \Delta t_k^j, \tag{1}$$

where D_k^j is the amount of tracking data generated by one camera per time unit and M is the number of activated cameras when tracking the j-th target. Let indicator $x_k^j = 1$ when edge S selects the k-th edge as one of j's tracking edges in its schedule. The computation overhead HyMOT needs to minimize when tracking target set $\mathbf{J}$ among edge set $\mathbf{E}$ is:

$$W = \sum_{j=1}^{|\mathbf{J}|} \sum_{k=1}^{|\mathbf{E}|} x_k^j \cdot l_k^j. \tag{2}$$

Intuitively asking all $|\mathbf{E}|$ edges to activate all of their cameras for tracking incurs prohibitive overheads in hyper-scale cameras. Based on the observations on bad angles, overlapping, and unnecessary tracking, we note that a target's movement trajectory is limited to a certain tempo-spatial pattern, which provides opportunities to effectively reduce the searching space. Specifically, these movement patterns could be elaborated from objects' historical movement statistics.

Spatial Correlation: To yield fine-grained spatial movement pattern discovery, we first introduce two spatial levels of cameras' monitored areas:

(Definition 1, Field) The joint surveillance region is covered by a cluster of nearby cameras that share surveillance intersections and are controlled by the same edge server.

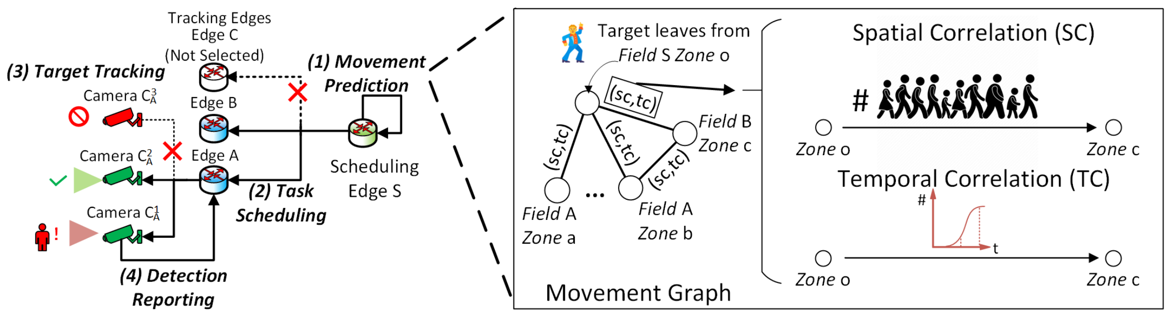


Fig. 2. **HyMOT's 4-stage workflow based on a hierarchical structure of scheduling edge, tracking edges, and cameras.**

(Definition 2, Zone) The regions that a target may enter or leave a *Field*.

As shown in Fig. 1, a field consists of several zones, each corresponding to one moving direction in the covered roads. Objects move between zones, so their spatial movement pattern lies in the activity of links between zones in different fields, which we call spatial correlation. For this, we build a **movement graph** (shown in Fig. 3) with zones as its nodes, connections as the edges, and correlations as the weights. If we have monitored an object reach *Field* A *Zone* a from *Field* B *Zone* b, then these two zones are connected in the graph.

W.l.o.g., assuming HyMOT receives a tracking task with target j leaving *Field* S *Zone* o. If $\mathbf{Z}_o$ is the group of zones that are connected with *Zone* o in the graph, then the spatial correlation between *Zone* o and zone $m \in \mathbf{Z}_o$ is estimated based on their inter-link's historical connection possibility:

$$p_{sc}^{So\rightarrow m} = \frac{Count(So,m)}{Count(So,exit) + \sum_{i\in\mathbf{Z}_o} Count(So,i)}, \tag{3}$$

where $Count(So,i)$ is the number of objects that have been monitored to leave *Field* S *Zone* o for *Zone* i, while $Count(So,exit)$ denotes the number of objects that leave from *Field* S *Zone* o but do not enter any surveillance regions anymore. Denote the possibility of targets' historically losing surveillance from *Zone* o as p_{So}^{exit}, we can have $p_{sc}^{So\rightarrow exit} + \sum_{m=0}^{|\mathbf{Z}_o|} p_{sc}^{So\rightarrow m} = 1$. As shown in the example in Fig. 2, spatial correlations between *Zone* o and Edge A/B/C are calculated and rendered in the form of heatmap.

Temporal Correlation: If objects usually leave *Zone* a and enter *Zone* b (in different fields) quickly, the latter should be assigned with tracking task preferentially during scheduling. Hence, for two connected zones in the movement graph, we quantify such temporal correlations by measuring their historical trip time. Denote $Count(a,b,-)$ as the number of objects going from *Zone* a to *Zone* b during historical surveillance and $Count(a,b,\Delta t)$ as those moving within time interval Δt. Then the temporal correlation can be estimated as:

$$p_{tc}^{a,b,\Delta t} = \frac{Count(a,b,\Delta t)}{Count(a,b,-)}. \tag{4}$$

Camera Selection: Surveillance cameras unavoidably have detection defects for working under diverse contexts and imperfect algorithms. This may cause *false positive* situations where the target passes the camera in the monitoring interval while not being detected, as well as *false negative* situations where a camera reports a detection on a target that has not entered its zones. We denote the c-th camera's empirical detection accuracy as p_{acc}^c. If totally N cameras are activated in *Field* F, then the overall reliability of *Field* F in this tracking task can be estimated with:

$$p_{re}^F = 1 - \prod_{c=1}^{N}(1 - p_{acc}^c). \tag{5}$$

Tracking optimization: Based on the above formulation, the possibility for HyMOT's scheduling to successfully track target j is given as:

$$p_{track}^j = \sum_{k=1}^{|\mathbf{E}|} x_k^j \cdot \sum_{F_k^i \in Fieldof(k)} x_{F_k^i}^j \cdot p_{sc}^{So\rightarrow F_k^i p} \cdot p_{tc}^{o,p,\Delta t} \cdot p_{re}^{F_k^i}, \tag{6}$$

where F_k^i represents a field under the k-th edge and $x_{F_k^i}^j$ is an indicator variable that equals 1 if cameras in F_k^i are selected for tracking in the schedule. Obviously, assigning tasks to fewer cameras could reduce computations, but at the cost of tracking failures. To maintain sufficient effectiveness, HyMOT requires the probability of losing target to retain in a controlled threshold:

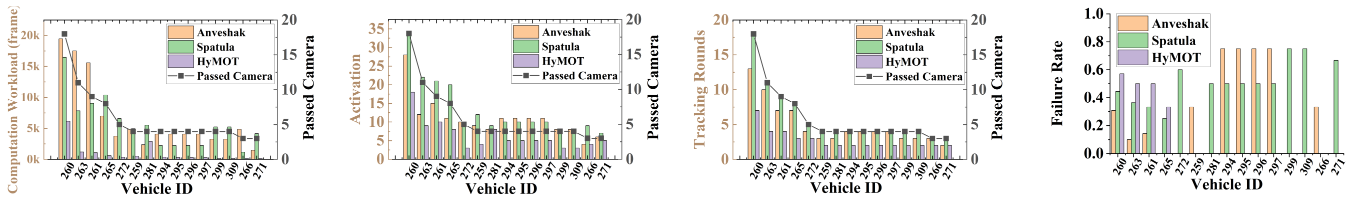
$$1 - p_{track}^j - p_{exit} < p_{exit} \cdot 10^{v_j}. \tag{7}$$

v_j is the predefined importance of target j, given by the stakeholder issuing the tracking task. Finally, the optimization problem for hyper-scale tracking efficiency in HyMOT can be formulated as scheduling the indicator variables:

$$\underset{x_k^j, x_{F_k^i}^j}{Minimize} \quad \sum_{j\in}^{|\mathbf{J}|} \sum_{k=1}^{|\mathbf{E}|} x_k^j \cdot l_k^j, \tag{8a}$$

$$s.t. \quad Equs.(1),(3),(4),(5),(6),(7). \tag{8b}$$

This is a typical linear programming problem, which is proved to be NP-Hard in computation. We use a greedy strategy to solve it, that is, for each issued tracking, unselected



(a) Computation Workload (b) Activation Number (c) Tracking Rounds (d) Failure Rate

Fig. 3. Evaluation results of three tracking scheduling methods on Cityflow.

Zone p with the largest spatial correlation with the target's leaving *Zone* is added to the scheduling pool (i.e., $x_k^j = 1$ and $x_{F_k^i}^j = 1$ with p a *Zone* in F_k^i) one by one until the constraint in Equ. (7) is satisfied.

4. EXPERIMENTS

4.1. Methodology

Baselines: For comparison, we use the direction-aware scheduling method Anveshak [19] and coarse-grained spatial correlation-based scheduling method Spatula [14].

Surveillance simulation: We simulate small-scale camera deployment and large-scale deployment based on datasets *Cityflow* and *Geolife*, respectively. For *Cityflow*, we estimate tempo-spatial correlations with scenario S05 and test performance with scenario S04. Totally 7 *Fields* are constructed for S04, each containing at least 1 camera and 1 *Zone*. For *Geolife Dataset*, we pick 10561 labeled tracklets from it for testing. We simulate a city-scale of 100 *Fields* with 4 *Zones* and 1-5 cameras (randomly selected) in each *Field*. 25% of the trajectory data is used to build the table in Spatula and movement graph in HyMOT, and the rest is used for tracking evaluation (i.e., around 8000 targets).

Metrics: We measure tracking efficiency using *Computation Workload* (i.e., the number of processed frames), *Activations* (i.e., the number of activated cameras during tracking), and *Tracking Rounds* (i.e., the number of rounds needed for performing tracking tasks). We measure the tracking accuracy with *Failure Rate*, which is the fraction of detection failures. Please note that Video analytics use deep models with fixed resources. By reducing processed frames, HyMOT saves both computation and storage resources.

4.2. Evaluation results

We compare the performance of HyMOT, Spatula, and Anveshak using Cityflow dataset. As shown in Fig.3, HyMOT outperforms the other two strategies in terms of computation workload, number of activated cameras, and tracking rounds, demonstrating superior efficiency for small-scale surveillance scenarios. Fig.3(d) shows that HyMOT has a higher failure rate for targets passing through more than 8 cameras, but achieves 100% accuracy for targets appearing in less than 5 cameras. This is because the amount of data used to construct spatial-temporal correlations are relatively small, which limits the correlation statistics needed for long-range tracking.

Table 1. Three methods' performance on Geolife.

	Computation Workload (frames)	Maximum activations	Tracking rounds	Average failure rate
Spatula	6.696×10^9	85	27	0.147
Anveshak	4.28×10^9	78	26	0.162
HyMOT	0.383×10^9	60	21	0.140

We then perform a large-scale evaluation to compare the performance of different methods in hyper-scale tracking. As shown in Table 1, HyMOT outperforms Spatula and Anveshak, requiring 94% and 91% fewer computations, respectively, to track around 8000 targets. HyMOT also achieves a similar or even smaller failure rate than the baselines while requiring fewer camera activations and tracking rounds. These results demonstrate the high feasibility of HyMOT in practical deployment due to its efficiency in tracking objects.

5. CONCLUSION

This work summarizes the issues that hinder the efficiency of target tracking in hyper-scale camera networks. A new tracking framework HyMOT is proposed by carefully handling the redundancy and over-activation in existing tracking scheduling strategies. HyMOT accommodates the tracking tasks in a finer grain and profiles the tempo-spatial correlation for target movement prediction, together reducing the edge servers' computation burden. Experimental results show that compared with the SoTA tracking strategies, HyMOT only requires 90% fewer computation resources for tracking the same amount of targets with similar tracking accuracy.

Acknowledgment

This work is supported by National Natural Science Foundation of China (62072465, 62102425), Science and Technology Innovation Program of Hunan Province (2021RC2071, 2022RC3061), and NUDT Research Grants (ZK19-38).

6. REFERENCES

[1] Hao Wu, Xinxiang Zhang, Brett Story, and Dinesh Rajan, "Accurate vehicle detection using multi-camera data fusion and machine learning," in *Proceedings of ICASSP*. IEEE, 2019, pp. 3767–3771.

[2] Chong Liu, Yuqi Zhang, Hao Luo, Jiasheng Tang, Weihua Chen, Xianzhe Xu, Fan Wang, Hao Li, and Yi-Dong Shen, "City-scale multi-camera vehicle tracking guided by crossroad zones," in *Proceedings of CVPR Workshop*, 2021.

[3] Yuanyuan Xu, Taoyu Yang, Zengjie Tan, and Haolun Lan, "Fov-based coding optimization for 360-degree virtual reality videos," in *Proceedings of ICASSP*, 2022, pp. 1640–1644.

[4] Ganesh Ananthanarayanan, Paramvir Bahl, Peter Bodík, Krishna Chintalapudi, Matthai Philipose, Lenin Ravindranath, and Sudipta Sinha, "Real-time video analytics: The killer app for edge computing," *Computer*, vol. 50, no. 10, pp. 58–67, 2017.

[5] Yuhang He, Xing Wei, Xiaopeng Hong, Weiwei Shi, and Yihong Gong, "Multi-target multi-camera tracking by tracklet-to-target assignment," *IEEE Transactions on Image Processing*, vol. 29, pp. 5191–5205, 2020.

[6] Haohua Du, Linlin Chen, Jianwei Qian, Jiahui Hou, Taeho Jung, and Xiang-Yang Li, "Patronus: A system for privacy-preserving cloud video surveillance," *IEEE Journal on Selected Areas in Communications*, vol. 38, pp. 1252–1261, 2020.

[7] Jiaping Yu, Haiwen Chen, Kui Wu, Tongqing Zhou, Zhiping Cai, and Fang Liu, "Centipede: Leveraging the distributed camera crowd for cooperative video data storage," *IEEE Internet of Things Journal*, vol. 8, no. 22, pp. 16498–16509, 2021.

[8] Jiaping Yu, Haiwen Chen, Kui Wu, Tongqing Zhou, Zhiping Cai, and Fang Liu, "Evichain: A scalable blockchain for accountable intelligent surveillance systems," *International Journal of Intelligent Systems*, vol. 37, no. 2, pp. 1454–1478, 2022.

[9] BriefCam, "Transforming video into actionable intelligence," `https://www.briefcam.com/`, 2022.

[10] Zheng Tang, Milind Naphade, Ming-Yu Liu, Xiaodong Yang, Stan Birchfield, Shuo Wang, Ratnesh Kumar, David Anastasiu, and Jenq-Neng Hwang, "Cityflow: A city-scale benchmark for multi-target multi-camera vehicle tracking and re-identification," in *Proceedings of CVPR*, 2019, pp. 8797–8806.

[11] Pan Hu, Junha Im, Zain Asgar, and Sachin Katti, "Starfish: Resilient image compression for aiot cameras," in *Proceedings of Sensys*, 2020, pp. 395–408.

[12] Mi Zhang, Faen Zhang, Nicholas D Lane, Yuanchao Shu, Xiao Zeng, Biyi Fang, Shen Yan, and Hui Xu, "Deep learning in the era of edge computing: Challenges and opportunities," *Fog Computing: Theory and Practice*, pp. 67–78, 2020.

[13] Sadid Sahami, Gene Cheung, and Chia-Wen Lin, "Fast graph sampling for short video summarization using gershgorin disc alignment," in *Proceedings of ICASSP*, 2022, pp. 1765–1769.

[14] Samvit Jain, Xun Zhang, Yuhao Zhou, Ganesh Ananthanarayanan, Junchen Jiang, Yuanchao Shu, Paramvir Bahl, and Joseph Gonzalez, "Spatula: Efficient cross-camera video analytics on large camera networks," in *Proceedings of SEC*. IEEE, 2020, pp. 110–124.

[15] Microsoft Azure, "Cloud computing services," `https://azure.microsoft.com`, 2022.

[16] Jiahui Liu, Mingcai Zhou, and Meng Xiao, "Deformable convolution dense network for compressed video quality enhancement," in *Proceedings of ICASSP*, 2022, pp. 1930–1934.

[17] Xiao Zeng, Biyi Fang, Haichen Shen, and Mi Zhang, "Distream: scaling live video analytics with workload-adaptive distributed edge intelligence," in *Proceedings of SenSys*, 2020.

[18] Daniel Angley, Sofia Suvorova, Branko Ristic, William Moran, Fiona Fletcher, H Gaetjens, and Sergey Simakov, "Sensor scheduling for target tracking in large multistatic sonobuoy fields," in *Proceedings of ICASSP*. IEEE, 2017, pp. 3146–3150.

[19] Aakash Khochare, Aravindhan Krishnan, and Yogesh Simmhan, "A scalable platform for distributed object tracking across a many-camera network," *IEEE Transactions on Parallel and Distributed Systems*, vol. 32, no. 6, pp. 1479–1493, 2021.

[20] Yu Zheng, Lizhu Zhang, Xing Xie, and Wei-Ying Ma, "Mining interesting locations and travel sequences from gps trajectories," in *Proceedings of WWW*, 2009, pp. 791–800.

[21] Yu Zheng, Quannan Li, Yukun Chen, Xing Xie, and Wei-Ying Ma, "Understanding mobility based on gps data," in *Proceedings of UbiComp*, 2008, pp. 312–321.

[22] Yu Zheng, Xing Xie, Wei-Ying Ma, et al., "Geolife: A collaborative social networking service among user, location and trajectory.," *IEEE Data Engineering Bulletin*, vol. 33, no. 2, pp. 32–39, 2010.