Cloud-based Real-time Outsourcing Localization for a Ground Mobile Robot in Large-scale Outdoor Environments

Xiaorui Zhu, Chunxin Qiu, Fucheng Deng, and Su Pang
State Key Laboratory of Robotics and System (HIT), Harbin Institute of Technology (Shenzhen), Shenzhen, Guangdong 518055, China
e-mail: fuchengdeng@hit.edu.cn
Yongsheng Ou
Shenzhen Institutes of Advanced Technology, Chinese Academy of Sciences, Shenzhen, Guangdong 518055, China

Received 4 May 2016; revised 9 January 2017; accepted 14 January 2017

Cloud robotics is the application of cloud computing concepts to robotic systems. It utilizes modern cloud computing infrastructure to distribute computing resources and datasets. Cloud-based real-time outsourcing localization architecture is proposed in this paper to allow a ground mobile robot to identify its location relative to a road network map and reference images in the cloud. An update of the road network map is executed in the cloud, as is the extraction of the robot-terrain inclination (RTI) model as well as reference image matching. A particle filter with a network-delay-compensation localization algorithm is executed on the mobile robot based on the local RTI model and the recognized location both of which are sent from the cloud. The proposed methods are tested in different challenging outdoor scenarios with a ground mobile robot equipped with minimal onboard hardware, where the longest trajectory was 13.1 km. Experimental results show that this method could be applicable to large-scale outdoor environments for autonomous robots in real time.

1. INTRODUCTION

Localization is an important problem for autonomous mobile robots. Robot localization, especially for large-scale explorations, has traditionally been limited by inherent physical constraints, since all the computations have to be conducted in the onboard computers/microchips of the robot with limited computing capabilities. In this paper, we propose a cloud-based architecture to achieve large-scale localization of mobile robots in outdoor environments, taking advantage of the powerful computation, storage, and other shared resources of the cloud.

In the past decade, many researchers have started to focus on robot localization in outdoor environments. Many outdoor environment localization algorithms depend on GPS. However, in some urban circumstances, e.g., an urban canyon, an alleyway, or a tunnel, the GPS signal is severely blocked, and the estimation error can reach more than 10 m in these situations (Bonnifait et al., 2001; Bouvet and Garcia, 2000; Drawil et al., 2013). Laser range scanners or sonar are options for autonomous outdoor navigation where a point cloud is generated to represent the surroundings (Brenneke et al., 2003; Fairfield et al., 2007; Hygounenc et al., 2004; Martin and Moravec, 1996; Nuchter et al., 2004; Pfaff et al., 2007; Triebel et al., 2006). However, point-cloud representation requires a high computational load (Brenneke et al., 2003). The combination of a point reduction and a kd-tree was proposed to reduce the computational load of point-cloud representation (Nuchter et al., 2004). Some researchers have used an occupancy grid map to divide a whole point cloud into a grid of cells with the occupancy evidence inferred from sensors (Martin and Moravec, 1996). Since the representation of the entire space must be stored in memory, even two-dimensional (2D) evidence grids are large and computationally expensive to copy. To reduce the computation and storage requirements, an octree data structure was further developed to finish an underwater tunnel exploration project using a simplified occupancy grid map (Fairfield et al., 2007). Hornung *et al.* developed an octree and probabilistic occupancy estimation based mapping approach to solve the problem of overconfident maps, and to keep the memory requirement to a minimum (Hornung et al., 2013). Alternatively, several researchers proposed to sort the raw point-cloud data by way of a standard elevation map (SEM) (Hygounenc et al., 2004) and a multilevel surface map (MLS) (Pfaff et al., 2007; Triebel et al., 2006). The application of vision systems for outdoor localization has also received

Author to whom all correspondence should be addressed: fuchengdeng@hit.edu.cn

increasing attention (Blaer and Allen, 2002). Different robust core algorithms, such as the scale-invariant feature transform (SIFT) and the position-invariant robust feature (PIRF), have been developed to adapt to complex outdoor environments (Kawewong et al., 2011; Valgren and Lilienthal, 2007). However, unpredictable long computation periods still made the above techniques fail for many real-time applications (Tongprasit et al., 2011). Outsourcing map-based localization is another method to estimate robot position. Christian *et al.* proposed a novel approach to take advantage of a road network structure and its height profile for position estimation when the GPS signal was lost (Mandel and Laue, 2010). Our research group also proposed a new localization method with less occupied memory where the topographical map was utilized as the prior available terrain map for localization (Zhu et al., 2013, 2014). However, these methods may fail when the robot encounters ambiguous routes with parallel roads. Xu *et al.* presented an outsourcing map-based algorithm with vision and road curvature estimation for localizing a vehicle to solve the ambiguous route problem (Xu et al., 2014). Recently, Majdik *et al.* proposed an air-ground image-matching algorithm to localize and track the pose of a microaerial vehicle (MAV) flying in urban streets without GPS (Majdik et al, 2015). This method also took outsourcing data such as ground-level street view images for localization where these images have been previously back-projected onto the cadastral 3D city model. However, their method is limited by high computational complexity, and it cannot be readily used in real-time applications (Majdik et al, 2015).

However, in the case of large-scale exploration, all of the above efforts are insufficient. Mobile robots such as rescue, surveillance, and military robots are often required to work in large-scale environments. A large-scale environment is defined as one whose spatial structure is at a significantly larger scale than the sensory horizon of the observer (Kuipers and Yung-Tai, 1991). Large-scale localization is challenging because of the demand for expensive computational power and huge memory storage due to the large amount of data to be processed (Vandapel et al., 2006). For example, the Stanley and Boss that won the DARPA Grand and Urban Challenge, respectively, were equipped with an array of 6–10 computers, and over 1 TB hard drive storage room was needed for data processing and logging (Thrun et al., 2006; Urmson et al., 2008). To reduce the memory requirement, Lankenau *et al.* presented a new algorithm called RouteLoc for the absolute self-localization of mobile robots in a large-scale environment (Lankenau and Rofer, 2002) represented by a hybrid topological-metric map. However, this method can be only used in structured environments. To deal with the complexity of an unstructured large-scale environment, Miguel *et al.* proposed a scalable machine learning approach to match the similarity of sequent images over long trajectories for robot localization with vision sensors (Miguel and Bonev, 2010). Bradley *et al.* presented a new method using "weighted gradient orientation histograms" as the features extracted from the images and estimating the robot position by matching these features (Bradley et al., 2005). This method could effectively reduce computational consumption in large outdoor environments. However, it was hard to discriminate the visually similar locations and match the features under the extreme illumination changes (Valgren and Lilienthal 2010). Xie *et al.* introduced a coarse-to-fine matching method for global localization, and they examined the problem of a mobile robot with a laser sensor in large-scale unstructured outdoor environments (Xie et al., 2010). However, the proposed localization approach would fail if the environments changed significantly or the onboard laser sensors were "blocked" by the surrounding crowd.

Recently, a few researchers have tried to overcome the challenges of a long-term autonomous robot in outdoor environments, where the mobile robot is expected to run autonomously over a long period and to adapt to real dynamic scenarios. Zhao *et al.* proposed a simultaneous localization and mapping (SLAM) method to simultaneously detect and track the moving objects using a laser scanner in a dynamic environment (Zhao et al., 2008). The authors found out that the method was very time-consuming to track many static or moving objects. Badino *et al.* and Sunderhauf *et al.* described a novel concept of an appearance change prediction to learn how the environment changes over time beforehand, and then to take advantage of the learned knowledge to predict its appearance under different environmental conditions (Badino et al., 2012; Sünderhauf et al., 2013). The key limitation of this method was the requirement for a large storage space to store different environmental conditions and the map information of the navigation area.

Cloud robotics, a concept first introduced by Kuffner at Google, provides a very promising solution to overcome the above problems faced by large-scale or long-term autonomous robots (Goldberg and Kehoe, 2013; Kuffner, 2010). Cloud robotics applies cloud-computing concepts to a robot in order to augment the capabilities of the robot by off-loading computation and sharing the huge amount of data or new skills via the Internet. Only a few papers on cloud robotics have been reported so far. Such papers include those on cloud-based SLAM, object grasping, and multirobotics. To tackle the challenges of the intensive mapping data and computation load during SLAM, Arumugam *et al.* built a cloud-computing infrastructure "DAvinCi" to improve the implementation speed of the SLAM algorithm (Arumugam et al., 2010). Riazuelo *et al.* described a visual SLAM system based on the cloud where the expensive map optimization and storage were allocated in the cloud, and a light camera tracking client ran on the onboard computer of the robot (Riazuelo et al., 2014). Kehoe et al. developed an architecture of a cloud robotics system to recognize and grasp common household objects (Kehoe et al., 2013). To

reduce the demand of manually labeled training data, Lai and Fox used objects from Google's 3D warehouse to help classify a 3D point cloud collected by a robot (Lai and Fox, 2010). Multiple robot cooperation is another area that could benefit from cloud robotics (Hu et al., 2012; Wang et al., 2012). Hu *et al.* proposed a cloud robotics architecture to handle the constraints faced by networked robots, and they described some specific challenges and potential applications in cloud robotics (Hu et al., 2012). To improve efficiency and share data among different robots, Wang *et al.* introduced a generic infrastructure for a cloud robotic system to enable several poorly equipped robots to retrieve location data from a dynamically updated map built by a well-equipped robot (Wang et al., 2012).

In particular, a cloud-based approach is important to robot localization of large-scale environments. The first need for a cloud-based approach concerns usage of a map. There is no doubt that a map could provide much more ample information to facilitate accurate and stable localization of large-scale environments. However, storage capacity of most mobile robots does not allow them to store all the map data of a very large environment onboard. In urban areas with a high road network density (e.g., 8 km/km^2 in China, about 20 km/ km^2 if side pavements and footways are included), it usually takes more than one gigabyte of storage capacity to cover an area only about 6 km^2. Therefore, it is unrealistic for many robots to store all the map data of a very large-scale environment, such as a whole city. On the other hand, if the real environment changes, it is hard to update the map in time if all the map data are only stored onboard. The second emphasis is that a cloud-based solution makes it possible to greatly lower the requirements of a robot platform. For instance, low-cost robots are able to accomplish complex missions collaboratively with a cloud-based framework, even for those robots designed with a low-cost embedded solution such as a smartphone-level configuration (Barbosa et al, 2015; Mohanarajah et al., 2015). Therefore, a cloud-based framework is expected to enable a low-cost robot to navigate in a large-scale challenging environment over a long period without greatly sacrificing robot performance.

In this paper, we propose a new cloud-based localization architecture using outsourced road network maps and reference images in the cloud to achieve real-time autonomous navigation of a mobile robot in large-scale outdoor environments. The road network maps and reference images are first extracted by Google Earth, OpenStreet Map, or other commercially available resources such that new road information could be added into the database in the cloud. The terrain inclination based localization algorithms previously proposed by our research group (Zhu et al., 2014) will then be extended to achieve large-scale online localization only taking advantage of the road network maps stored in the cloud. A vision-based supplement is proposed to improve performance further using the reference images stored in the cloud. The main contributions of this paper are threefold:

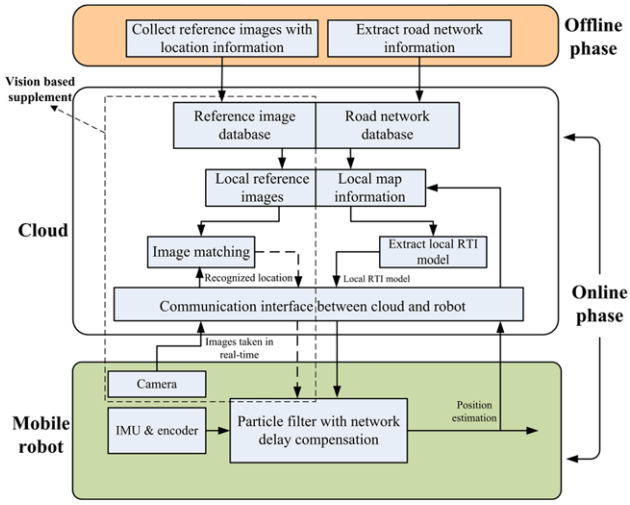

Figure 1. Proposed cloud-based localization architecture.

- In the proposed architecture, a ground mobile robot can achieve accurate real-time localization in different large-scale challenging scenarios where the GPS signal is often shadowed or completely unavailable.
- The proposed methodology allows the mobile robot, which is only equipped with minimal hardware and onboard sensors, to access a large amount of data stored in the cloud. On the other hand, network delay caused by data interchanging between the cloud and the robot is compensated for successfully.
- A vision-based supplement technique is used to effectively solve the problem of location ambiguity resulting from the accumulative error of onboard sensors when the robot travels along a very long, straight, and flat road segment. The proposed technique guarantees a limited localization error within 0.5 m during the entire long-term operation.

The remainder of the paper is organized as follows. The cloud robotics architecture and detailed algorithms are proposed in Sections 2 and 3, respectively. Experimental results and discussion are presented in Section 4. Conclusions are described in Section 5.

2. CLOUD-BASED OUTSOURCING LOCALIZATION ARCHITECTURE

The proposed cloud-based architecture has two phases: offline and online; see Figure 1. Each phase will be illustrated in detail in the following subsections.

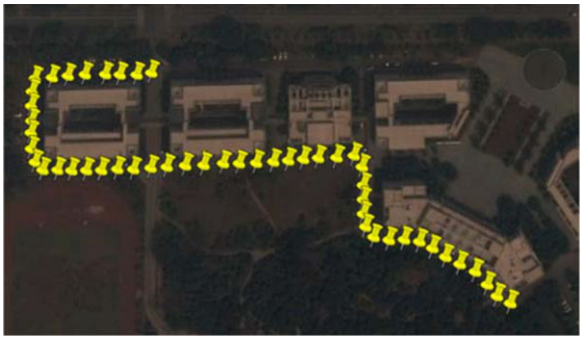

Figure 2. Extraction of point set on a preplanned path.

2.1. Offline Phase

The offline phase consists of two parts: the extraction of road network information and the construction of reference images. The former is to extract new road networks and update the existing road network map stored in the cloud. It is assumed that the robot only travels on the road network. To build the road networks, first a set of reference points are labeled along the new road. Then the geodetic coordinates (latitude, longitude, altitude) of these points together with the road name can be extracted from Google Earth or other Geographic Information System (GIS) software, as Figure 2 depicts. A higher density of these reference points would result in higher localization accuracy. Finally, the latest road information is integrated into the road network database, which is stored and maintained in the cloud.

To improve the localization accuracy for mobile robots further, a reference image database is constructed in the proposed framework. An image of a specific location in the road network is taken as a reference for localization. Each image is tagged with location information *a priori*. To increase the accuracy and robustness of image matching, the reference images are always taken with salient features that are easily identifiable from the open source GIS software (e.g., Google Earth and OpenStreet Map), such as an image of a building with sharp edges, ground with obvious markers or color shift, etc.; see Figure 3. The reference images could also be taken manually if the area is not covered by the existing database. To obtain sufficient textures, it is preferred to take images of the lateral side of the road with an elevation angle of about 10–20°. These images will be updated and integrated into the reference image database stored in the cloud as well.

2.2. Online Phase

The online phase is for mobile robot localization based on the cloud. According to Figure 1, the architecture consists of two ends: the cloud and the mobile robot. On the cloud end, the robot terrain inclination (RTI) model (Zhu et al., 2014) can be extracted from the road network map and used to describe the relationship between the robot attitude and the robot position. On the other hand, image matching between an image taken by a mobile robot in real-time and the local reference images is also implemented on the cloud for a vision-based supplement to improve localization performance. An interface is designed for all the communication between the cloud and the robot. On the mobile robot end, onboard sensors [inertial measurement unit (IMU), encoder, and camera] are in place, and a particle filter based localization algorithm with network delay compensation is used to estimate the robot position.

The whole procedure is as follows. When the robot moves on a road, the initial position estimated by GPS will be sent to the cloud. All the local road information and reference images within an area of a radius δ centered at the initial position are automatically searched from the road network database and the reference image database, respectively, on the cloud. The RTI model for the corresponding road segments is extracted. Then the local RTI model for the road network is sent back to the robot. At the same time, images are received sequentially from the moving mobile robot and compared with the local reference images on the cloud. Once image matching is successful, the corresponding geodetic information of the reference image will be sent back to the robot. Finally, the robot will fuse the above local RTI model and the recognized location from the vision-based supplement with the motion model in a particle filter to estimate the current robot position. Notice that network delay is fully considered in the particle filter algorithm. Hence, in the proposed cloud-based architecture, most computation load as well as storage load can be distributed on the cloud such that the onboard microchip/computer of the robot will not be overloaded even when the exploration area becomes larger and larger.

3. CLOUD-BASED LOCALIZATION ALGORITHMS

3.1 Algorithms on the Cloud

3.1.1 Robot-terrain Inclination Model

The Cartesian coordinates (x, y, z) in this paper are defined as the local East, North, Up (ENU) coordinates at the estimated initial position of the robot. The y axis always points to the north-pole of the earth and the z axis points upward away from the earth's center. Suppose there is one portion of the road networks $B_1A_1B_jA_j$. As Figure 4(a) shows, the positions of the point set on the road are first transformed from the geodetic coordinates to the Cartesian coordinates (x, y, z). They are then projected onto the x-O-y plane as $B'_1A'_1B'_jA'_j$. The projected road is segmented into a series of line segments with a fixed interval L/n, where L is the length of the robot, and the parameter n can be adjusted for the accuracy requirement. $B'_jA'_j$ represents the jth line segment. The points B'_j and A'_j are the projections of the road points B_j and A_j, respectively. The z value of these road

Figure 3. Some examples of reference images along the road by (a) a building with sharp edge, (b) ground with distinct boundary and markers, (c) a unique statue or object, or (d) a crossroad with many textures.

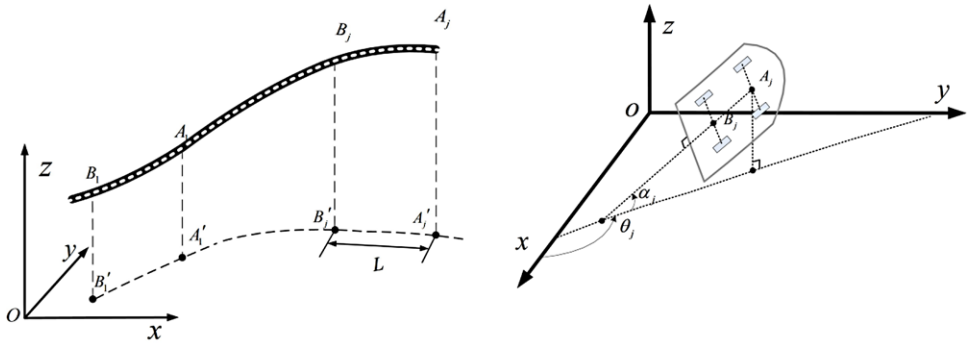

Figure 4. Robot-terrain inclination (RTI) model: (a) robot path segmentation for one portion of the road networks, (b) geometric extraction at one position.

points can be obtained by a weighted average interpolation method from the road network map (Chen et al., 2007). When the robot moves above the jth line segment along the preplanned path, the points B_j and A_j represent the midpoint between two ground contact points of the front wheels and the left-rear wheel, respectively.

Figure 4(b) shows the geometric extraction of the RTI model at one position. The $\overrightarrow{B_jA_j}$ represents the direction of the robot motion. The heading angle θ_j is defined as the angle between $\overrightarrow{B'_jA'_j}$ and the x axis. The heading angle is exclusively determined by the path. The angle α_j is defined as the one between the robot direction $\overrightarrow{B_jA_j}$ and the x-O-y plane. So the angles α_j can be obtained from the following equations:

$$\alpha_j = \sin^{-1}((z_A - z_B)/\left|\overrightarrow{B_jA_j}\right|), \tag{1}$$

$$\left|\overrightarrow{B_jA_j}\right| = \sqrt{(x_B - x_A)^2 + (y_B - y_A)^2 + (z_B - z_A)^2}, \tag{2}$$

where $B_j = (x_B, y_B, z_B)$ and $A_j = (x_A, y_A, z_A)$. Therefore, a number of angle pairs (θ_j, α_j) can be extracted from the serial line segments $B'_jA'_j$. Then the robot position (x_j, y_j, z_j) at each line segment corresponds with each group (θ_j, α_j). By linear interpolation of the above discrete relationship, $[\theta_k \ \alpha_k]^T = RTI(x_k, y_k, z_k)$ can be obtained, $k = 1, 2, \ldots, N$, where the number N can be adjusted for accuracy.

3.1.2. Image Matching

In the reference image database, each image is linked to a specific geodetic coordinate in the map. An image taken by the mobile robot in real-time is matched with reference images stored in the cloud. Once there is a successful match, the corresponding coordinates will be sent to the robot to help update the position estimation. The matching process is based on the Speed Up Robust Features (SURF) (Bay et al, 2008). A U-SURF detector/descriptor is used to extract key points from images and compute the 64-dimensional non-rotation-invariant descriptors for these regions. Note that non-rotation-invariant descriptors can reduce the matching time remarkably and enable real-time application. In this paper, the matching process is done by two directions. That is to say, for each feature point in a matching image, two closest neighbors should be found in the reference image. Similarly, two closest neighbors should be found in the matching image as well for a feature point in the reference image. A mutual ratio is defined as the ratio of the distance between the feature point in a matching/reference image and the closest neighbor in the reference/matching image over the one for the next closest one. If the mutual ratio is greater than a given threshold, then the closest neighbor point can

be accepted as the good candidate. But if the mutual ratio is lower than the threshold, both neighbor points are rejected to avoid wrong matching, which has made a large number of unreliable matches to be removed. Then two good match sets are obtained: one is from the matching image to the reference image, while the other is from the reference image to the matching image. A match pair is accepted only if these two match sets are in one-to-one correspondence. Finally, the Random Sample Consensus (RANSAC) method is used to compute the fundamental matrix, which is used to check whether the matches satisfy the epipolar constraint. Once a successful match is found, the location is recognized and the corresponding geodetic coordinate will be sent to the mobile robot.

3.2. Localization Algorithm on the Robot

3.2.1 Particle Filter Based Localization

In this paper, a particle filter based localization algorithm is designed on the robot to estimate the position. The detailed algorithm is summarized in Algorithm 1. The system state X_{t-1} represents the three-dimensional position of the robot in the inertial Cartesian frame (x, y, z) at the time t-1. Q_t is the covariance matrix of the system. Z_t is the system measurement vector. The IMU and encoder measurements from time t-k-1 to t are stored in Z_{backup} and v_{backup}, respectively, where k stands for the network time delay caused by data exchange between the robot and the cloud. Z_{backup}, v_{backup}, and X_{t-k-1} are used for network delay compensation, which will be presented in the following subsection. An index VI (Line 3, Algorithm 1) is used to indicate whether the robot receives the recognized location result from the cloud. When VI is equal to 0 (i.e., not receiving feedback from the cloud), the algorithm from Line 4 to Line 17 will be used to estimate robot position, X_t, at time t. Otherwise, a network delay compensation algorithm is proposed to be implemented additionally because a vision-based supplement would induce non-negligible network delay.

The motion model for one particle is shown in Line 5, Algorithm 1, where the superscript $[m]$ denotes the particle m. M is the total number of the particles. T is the sampling period, and v_t is the linear velocity in the direction of the robot movement. The measurement prediction is depicted in Line 6, Algorithm 1, where RTI_θ and RTI_α are the RTI models downloaded from the cloud, which are treated as part of the measurement model. $\hat{d}_t^{[m]}$ is the Euclidean distance between the particle m and the point $p(x, y, z)$, where $p(x_t^{[m]}, y_t^{[m]}, z_t^{[m]})$ represents the position of the particle m while $p(x, y, z)$ represents the road point on the robot path closest to this particle that is gained from the map. $\hat{V}_t^{[m]}$ represents the measurement prediction of the robot position in the geodetic coordinates obtained from the cloud only in the case of successful image matching ($VI = 1$). Then the state update is depicted in Lines 7–17, where θ_t and α_t are the measurements of yaw and pitch obtained from the onboard inertial measurement unit at time t. The distance d_t is not a real sensor measurement but a virtual measurement to keep all samples along the path at time t (Mandel and Laue, 2010). V_t is the recognized location result from the cloud. Line 8 in Algorithm 1 indicates the weight calculation of each particle where $w_t^{[m]}$ is the weighting factor of the particle m.

It is worth mentioning that if the initial position estimated by the GPS is far from the real position, the robot has to relocalize itself. Moreover, when the robot moves along a very long straight and flat road without identifiable visual features for a vision-based supplement, the distribution of the particle sets might deviate from the real position. Such situations are similar to the robot kidnapping problem and could make the implementation of the particle filter fail (Engelson & McDermott, 1992). Hence, in order to adapt to the above situations, additional hypotheses generated from sensing are inserted for sensor-based resampling to the particle filter localization when wrong estimations are detected (Lenser and Veloso, 2000). This process is called sensor resetting (Line 16 in Algorithm 1). If the sum of the particle weight $\sum w_i$ is above a threshold K, all the samples are kept. Otherwise, wrong estimation detection is assumed, and the sensor-based resampling will start. By matching the sensor measurement with the RTI model in a certain range, a best matching position can be found. Then a fixed number of samples will be added after resampling.

3.2.2 The Network Delay Compensation

Assumption in this paper is that the cloud and robot are able to connect with each other through a wireless network. The cloud server creates a listener socket that is waiting for connections from remote clients. The client issues the connect() socket function to start the TCP handshake. This socket contains many parameters of the client, such as the IP address, the port number, and so on. If these parameters are the same as those in the listener socket, then the cloud server issues an accept() socket function to accept the connection request. Thus the communication between the cloud and the robot can be established. In this paper, network delay caused by data exchange could affect the real-time performance of robot localization, while data processing in the cloud is much less time-consuming and would not generate any apparent delay. Moreover, network delay would be larger as more data are exchanged between the cloud and the robot.

In the proposed architecture, two types of data would exchange between the cloud and the robot, i.e., the local RTI model and images in the vision-based supplement part. The network delay caused by the local RTI model is always negligible for robot localization because most data and algorithms are designed to be relocated into the cloud, and

Table Algorithm 1. Particle filter based localization.

1	$(X_{t-1}, Z_t, v_t, X_{t-k-1}, Z_{\text{backup}}, v_{\text{backup}})$
2	$X_{t-1} = \langle \chi_{t-1}^{[1]}, \chi_{t-1}^{[2]}, ..., \chi_{t-1}^{[M]} \rangle$, $Z_t = \{\theta_t, \alpha_t, d_t, V_t\}$, Q_t, $\overline{X}_t = X_t = \emptyset$, $X_{t-k-1} = \langle \chi_{t-k-1}^{[1]}, \chi_{t-k-1}^{[2]}, ..., \chi_{t-k-1}^{[M]} \rangle$, $Z_{\text{backup}} = \{Z_{t-k-1}, Z_{t-k}, ..., Z_t\}$, $v_{\text{backup}} = \{v_{t-k-1}, v_{t-k}, ..., v_t\}$.
3	if $VI = 0$ do
4	for $m = 1$ to M do
5	$\chi_t^{[m]} = \begin{bmatrix} x_t^{[m]} \\ y_t^{[m]} \\ z_t^{[m]} \end{bmatrix} = \begin{bmatrix} x_{t-1}^{[m]} + \cos\theta_{t-1} \cdot \cos\alpha_{t-1} \cdot v_{t-1} \cdot T \\ y_{t-1}^{[m]} + \sin\theta_{t-1} \cdot \cos\alpha_{t-1} \cdot v_{t-1} \cdot T \\ z_{t-1}^{[m]} + \sin\alpha_{t-1} \cdot v_{t-1} \cdot T \end{bmatrix}$ //sample new pose
6	$\hat{Z}_t^{[m]} = \begin{bmatrix} \hat{\theta}_t^{[m]} \\ \hat{\alpha}_t^{[m]} \\ \hat{d}_t^{[m]} \\ \hat{V}_t^{[m]} \end{bmatrix} = \begin{bmatrix} RTI_\theta(x_t^{[m]}, y_t^{[m]}, z_t^{[m]}) \\ RTI_\alpha(x_t^{[m]}, y_t^{[m]}, z_t^{[m]}) \\ \text{dist}[p(x, y, z), p(x_t^{[m]}, y_t^{[m]}, z_t^{[m]})] \\ \mathbf{0} \end{bmatrix}$ //measurement prediction
7	$Z_t - \hat{Z}_t^{[m]} = \begin{bmatrix} \theta_t - \hat{\theta}_t^{[m]} \\ \alpha_t - \hat{\alpha}_t^{[m]} \\ d_t - \hat{d}_t^{[m]} \\ V_t - \hat{V}_t^{[m]} \end{bmatrix}$
8	$w_t^{[m]} = \lvert 2\pi Q_t \rvert^{-\frac{1}{2}} \exp\left\{-\frac{1}{2}(Z_t - \hat{Z}_t^{[m]})^T\right\} Q_t^{-1}\left((Z_t - \hat{Z}_t^{[m]})\right)$, //weight calculation
9	add $\chi_t^{[m]}$ and $w_t^{[m]}$ to $\bar{X}_t$
10	endfor
11	if $\sum w_i > K$, for $m = 1$ to M do
12	draw i with probability $\propto w_t^{[i]}$
13	add $\chi_t^{[i]}$ to X_t
14	endfor
15	else
16	replace particles with more reliable ones //sensor resetting
17	endif
18	return $X_t = \langle \chi_t^{[1]}, \chi_t^{[2]}, ..., \chi_t^{[M]} \rangle$
19	if $VI = 1$ do
20	***network delay compensation***
21	endif

data exchange between the cloud and the robot has been minimized to reduce time delay within a range of negligibility. On the other hand, network delay in the vision-based supplement part has a significant effect on the localization because real-time image transmission from the robot to the cloud would result in non-negligible network delay. Hence, a network delay compensation algorithm is designed when VI is equal to 1 (Lines 19 and 20, Algorithm 1). The detailed algorithm is summarized in Algorithm 2.

Assume that the robot receives the recognized location result from the cloud at the current time t. Then it is inferred that this recognized location is associated with the image taken from the onboard camera of the robot at the time t-k. The main idea of the compensation is to take advantage of the backup sensor data during the delay time (the interval between the time when the image is taken and the time when the feedback is received from the cloud) to re-update the system state. When the robot receives the recognized location feedback from the cloud, another particle filter is used to fuse the feedback and the backup sensor data to re-update the estimated position of the robot, Algorithm 2. This algorithm is similar to Algorithm 1. The main difference is the introduction of a measurement function $IL[p(x_i^{[m]}, y_i^{[m]}, z_i^{[m]})]$ for the vision-based supplement (Line 5, Algorithm 2), which is the transformation function from the ENU Cartesian coordinates to the geodetic coordinates. The conversion can be done by the following two steps: 1) convert the local ENU coordinate to the Earth-centered

Table Algorithm 2. Network delay compensation.

1 $(X_{t-k-1}, Z_{\text{backup}}, v_{\text{backup}})$
2 for $i = t\text{-}k$ to t do //state re-update during network delay intervals
3 for $m = 1$ to M do
4 $\chi_i^{[m]} = \begin{bmatrix} x_i^{[m]} \\ y_i^{[m]} \\ z_i^{[m]} \end{bmatrix} = \begin{bmatrix} x_{i-1}^{[m]} + \cos\theta_{i-1} \cdot \cos\alpha_{i-1} \cdot v_{i-1} \cdot T \\ y_{i-1}^{[m]} + \sin\theta_{i-1} \cdot \cos\alpha_{i-1} \cdot v_{i-1} \cdot T \\ z_{i-1}^{[m]} + \sin\alpha_{i-1} \cdot v_{i-1} \cdot T \end{bmatrix}$
5 $\hat{Z}_i^{[m]} = \begin{bmatrix} \hat{\theta}_i^{[m]} \\ \hat{\alpha}_i^{[m]} \\ \hat{d}_i^{[m]} \\ \hat{V}_i^{[m]} \end{bmatrix} = \begin{bmatrix} RTI_\theta(x_i^{[m]}, y_i^{[m]}, z_i^{[m]}) \\ RTI_\alpha(x_i^{[m]}, y_i^{[m]}, z_i^{[m]}) \\ \text{dist}[p(x, y, z), p(x_i^{[m]}, y_i^{[m]}, z_i^{[m]})] \\ IL[p(x_i^{[m]}, y_i^{[m]}, z_i^{[m]})] \end{bmatrix}$
6 $Z_i - \hat{Z}_i^{[m]} = \begin{bmatrix} \theta_i - \hat{\theta}_i^{[m]} \\ \alpha_i - \hat{\alpha}_i^{[m]} \\ d_i - \hat{d}_i^{[m]} \\ V_i - \hat{V}_i^{[m]} \end{bmatrix}$
7 $w_i^{[m]} = |2\pi Q_i|^{-\frac{1}{2}} \exp\left\{-\frac{1}{2}(Z_i - \hat{Z}_i^{[m]})^T\right\} Q_i^{-1}\left((Z_i - \hat{Z}_i^{[m]})\right)$
8 add $\chi_i^{[m]}$ and $w_i^{[m]}$ to $\bar{X}_i$
9 endfor
10 for $m = 1$ to M do
11 draw j with probability $\propto w_i^{[j]}$
12 add $\chi_i^{[j]}$ to X_i
13 endfor
14 endfor
15 return $X_t = \langle \chi_t^{[1]}, \chi_t^{[2]}, ..., \chi_t^{[M]} \rangle$

Earth-fixed (ECEF) coordinate as

$$\begin{bmatrix} X_i^{[m]} \\ Y_i^{[m]} \\ Z_i^{[m]} \end{bmatrix} = \begin{bmatrix} -\sin\lambda_r & -\sin\varphi_r\cos\lambda_r & \cos\varphi_r\cos\lambda_r \\ cos\lambda_r & -\sin\varphi_r\sin\lambda_r & \cos\varphi_r\sin\lambda_r \\ 0 & \cos\varphi_r\sin\lambda_r & \sin\varphi_r \end{bmatrix} \begin{bmatrix} x_i^{[m]} \\ y_i^{[m]} \\ z_i^{[m]} \end{bmatrix} + \begin{bmatrix} X_r \\ Y_r \\ Z_r \end{bmatrix}, \quad (3)$$

where (X_r, Y_r, Z_r) represents the initial coordinate of the robot in the ECEF coordinate, and λ_r and φ_r represent the longitude and latitude, respectively. $(X_i^{[m]}, Y_i^{[m]}, Z_i^{[m]})$ represents the position of the particle m in the ECEF coordinates at the time i. 2) Convert the ECEF coordinate to the geodetic coordinate as

$$\lambda_i^{[m]} = \arctan(Y_i^{[m]} / X_i^{[m]}), \quad (4)$$

$$\varphi_i^{[m]} = arctan\left\{\frac{Z_i^{[m]} + be' \sin^3 U}{\sqrt{(X_i^{[m]})^2 + (Y_i^{[m]})^2} - ae^2\cos^3 U}\right\}, \quad (5)$$

$$h_i^{[m]} = Z_i^{[m]} / \sin\varphi_i^{[m]} - \text{N}(\varphi_i^{[m]})(1 - e^2), \quad (6)$$

where $\text{N}(\varphi_i^{[m]}) = a/\sqrt{1 - e^2\sin^2\varphi_i^{[m]}}$, $U = \arctan(Z_i^{[m]} a/\sqrt{(X_i^{[m]})^2 + (Y_i^{[m]})^2} b)$, and $b = \sqrt{(1 - e^2)a^2}$. The parameter a represents the semimajor axis of the ellipsoid, while e and e' are the first and second numerical eccentricity of the ellipsoid, respectively (Axes conventions, 2016). Time consumption of this compensation algorithm is relatively negligible compared with network delay because all sensor data in the previous few sampling intervals have been in the backup space automatically. Therefore, the side effect of network delay is efficiently eliminated while the network delay compensation would not introduce additional delay.

4. EXPERIMENTS AND DISCUSSIONS

4.1. Methods and Procedures

The experiments were conducted on a Summit XL® mobile robot platform with a smartphone-level onboard processor. The cloud service was run on a desktop PC (Intel Xeon CPU X5650 at 2.67 GHz, 4 GB RAM) offered by the National Super Computer Center in Shenzhen. The network was chosen as China Telecom's TD-LTE and FDD-LTE network where the theoretical uplink (from the subscriber to the Internet) peak rate was 50 Mbps and the downlink (from the Internet to the subscriber) peak rate was 100 Mbps. The NAV440 from Crossbow Technology® was used as the IMU that was mounted on the top surface of the robot in order to measure roll, pitch, and yaw angles. The measurement accuracy was 0.5° in the roll and pitch directions while it was 1° in the yaw direction. The linear velocity of the robot was provided by the encoder. A standard camera was also mounted on the top of the robot with an output of 640 × 480 images at 30 fps. Moreover, a standard commercial GPS module was integrated onboard only to provide an initial estimated position for the robot. (Note that this is not a requirement in the proposed approach, and an alternative method could be considered as long as the initial position is available for the robot.) The sampling period for all experiments was 0.1 s. In all experiments, the robot moved at a speed of less than 3 m/s and sent a request for a local map with a radius of 200 m when it traveled up to 100 m. Three outdoor scenarios have been selected near the Shenzhen University town for experimental validation. All the experiments were run in real-time using the wireless network. Two localization methods were used to do a comparison with the proposed method, such as a GPS-only method and a traditional particle filter based method incorporating GPS and inertial sensors without

using cloud service (refer to "Particle filter with GPS/INS" in the main section).

Scenario 1: An outdoor environment with an area of 400 m × 200 m was used for performance evaluation, as Figure 5(a) shows ($R1 \rightarrow R2 \rightarrow R3 \rightarrow R4$). The performance of localization is first tested in this scenario. The initial position error could be more than 10 m with the GPS. Thus, a series of experiments were conducted with different initial errors from 5 to 20 m to evaluate the robustness of the proposed method in terms of different initial position errors. A few artificial markers with the same interval (20 m) were labeled along the road. Then the positions of those markers were calculated individually from the road network map as references. Moreover, network disconnection between robot and cloud was also a phenomenon that was observed occasionally. The system performance under this situation was evaluated in this scenario as well.

Scenario 2: The experimental area spans approximately 700 m × 500 m, as Figure 5(b) shows. The robot traversed 1.8 km from points P1 to P8 covering several different types of roads, such as minor arteries, alleyways, and urban canyons. The experiments conducted in Scenario 2 were used to demonstrate that the proposed technique is applicable to the trips with complex road types.

Scenario 3: The third experimental area is about 2,500 m × 6,700 m. The robot traversed more than 13 km on nearly all kinds of road types, including an urban expressway, a primary artery, a minor artery, an alleyway, urban canyons, and tunnels. This experiment is used to demonstrate the performance of the proposed architecture in more general large-scale outdoor environments.

4.2. Results and Discussion

Scenario 1: The road networks of the selected area were extracted from Google Earth and sent to the cloud server before the localization task started. When comparing with the road network database saved previously [Figure 6(a)], it was found that Roads 1 and 2 on the current road networks were new [Figure 6(b)]. Therefore, the latest road network information was updated in the cloud.

When the robot started to move on the road, the initial position estimation $E1$ by GPS was transmitted to the cloud. The roads within an area of radius δ centered at point $E1$ were obtained, as Figure 6(b) shows. The initial position estimation may not be on the road due to the drift of GPS. In such a case, the nearest road point from $E1$ was chosen as the starting point of the RTI model. Then a local RTI model starting from this road point along the pre-planned path was computed in the cloud and sent back to the mobile robot. Finally, localization can be achieved on the robot by applying the particle filter algorithm in Section 3 (Algorithm 1 and Algorithm 2). When the robot traveled on road segments with a constant distance (200 m), the estimated robot position at the end of each segment [E2 and E3 in Figure 6(b)] was sent to the cloud again, and the above procedures were repeated continuously. Twenty-five discrete reference positions were used in this scenario.

Figure 7 shows the position estimation errors with different initial errors that were used to simulate different resolutions of GPS. The position estimation error Δd was defined as the 3D Euclidean distance between the estimated position and the reference position. Figures 7(a) and 7(b) show the localization performance without and with using sensor resetting, respectively. For the case without sensor resetting, it can be seen that the performance was acceptable only when the initial error provided by GPS was less than 5 m but the error diverged away when the initial errors were larger than 5 m. Conversely, with sensor resetting in the proposed algorithm, the position estimation error was reduced quickly even with an initial error of 20 m. Hence, the proposed method is robust to large initial position errors.

Figure 8 depicts the position estimation of the robot using the proposed method (solid line) compared with GPS-only (black dot), and the particle filter with GPS/INS (green triangle) and the reference positions (circles). Position estimation by the proposed method is much closer to the references. Area 1 (reference points $R1 \rightarrow R2$) and Area 2 (R3→R4) were surrounded by tall buildings, and the resolution of the GPS signals was instable. The same phenomena were also observed in Figure 9, which shows a comparison of the position estimation errors using the proposed method and the counterparts in the Cartesian coordinates. The estimation error at the traveling distance of 360 m was up to (9 m, 6 m) and (6 m, 2.3 m) using GPS only and a particle filter with GPS/INS, respectively, because GPS was partially blocked around this area. It was already somewhere offroad. In contrast, the estimation error using the proposed method was only (0.2 m, 0.3 m), which was consistent with the real situation.

The influence of network connection loss was also evaluated in Scenario 1. The robot will invoke a request for new local road information from the cloud only when the robot travels out of a constant circular area, i.e., a radius of 200 m. Thus, if the robot is moving within such an area, the disconnection of the network would not greatly influence the localization performance because terrain-based localization algorithms still work without a vision-based supplement. When the robot moves out of the area, dead reckoning is used to estimate the position with the IMU and the encoder module on the robot. Because of vibration and sensor drift, the position estimation error of dead reckoning increases with time (Figure 10). Once the robot reconnects to the cloud, it will send the previous estimated position to the cloud. Then the cloud will respond to the robot with the nearby road network information. The position estimation error caused by network disconnection could be treated as the initial error. According to the previous analysis, the proposed algorithm is robust to large initial errors. Hence, a

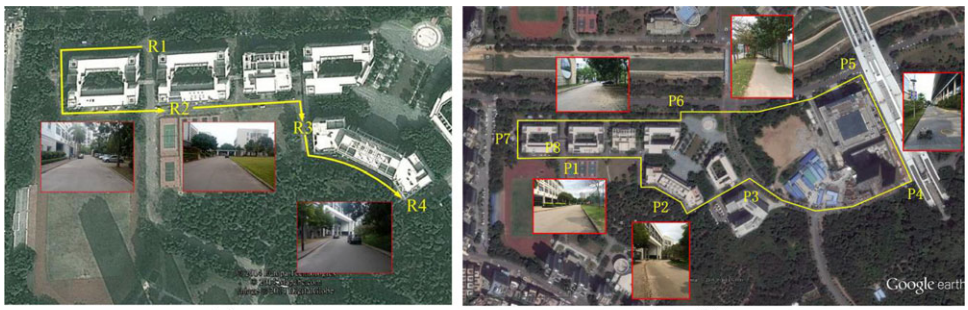

Figure 5. Pictures of experimental areas: (a) scenario 1, (b) scenario 2.

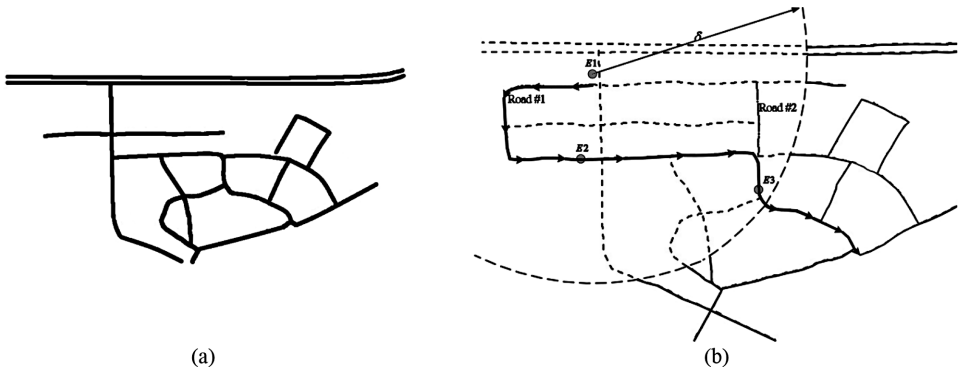

Figure 6. Road networks of a navigation area: (a) a previously existing road network, and (b) an updated road network (the road with arrows represents the planned robot path).

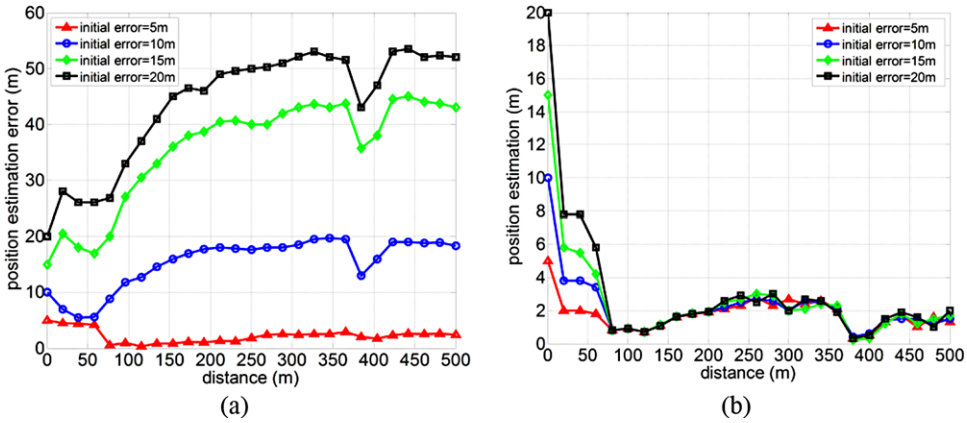

Figure 7. Position estimation errors with different initial errors (a) without sensor resetting, and (b) with sensor resetting.

short period of network loss would not affect the performance of the whole system.

Scenario 2: Figure 11 shows the comparison of position estimation with the proposed method (red solid line), GPS-only (blue dotted line), and particle filter with GPS/INS (yellow solid line) in a larger outdoor environment (Scenario 2). The robot traversed on several types of roads where the GPS signals were not always good. It can be seen that when the robot traversed on alleyways (C to D, 100 m) and urban canyons (G to H, 230 m), localization results deviated far from the references on the road because the GPS signal was seriously blocked by the tall buildings. The position estimation error by GPS in these two road segments was more than 30 m, while the maximum position error using the proposed method was about 2 m. It is also observed that the destination point estimated by the proposed method was

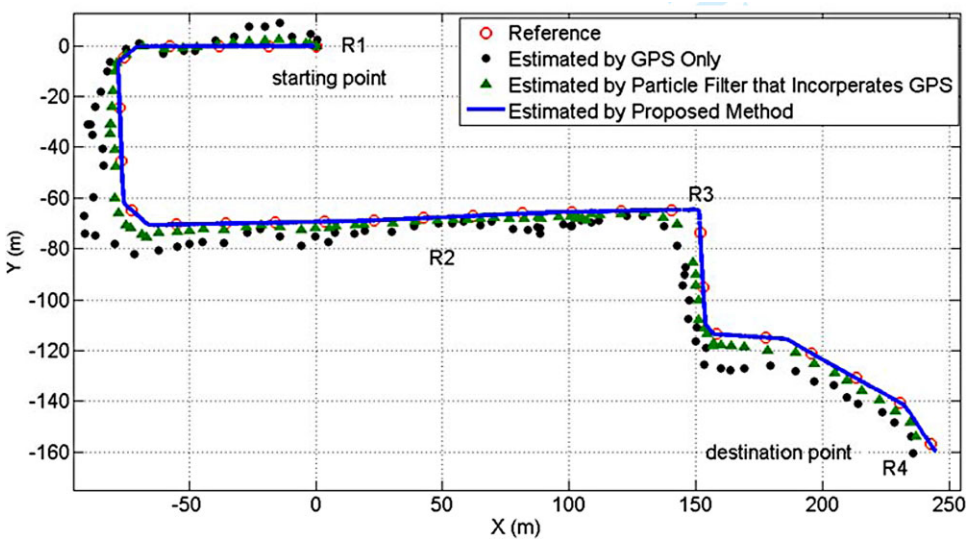

Figure 8. Comparison of position estimation (scenario 1).

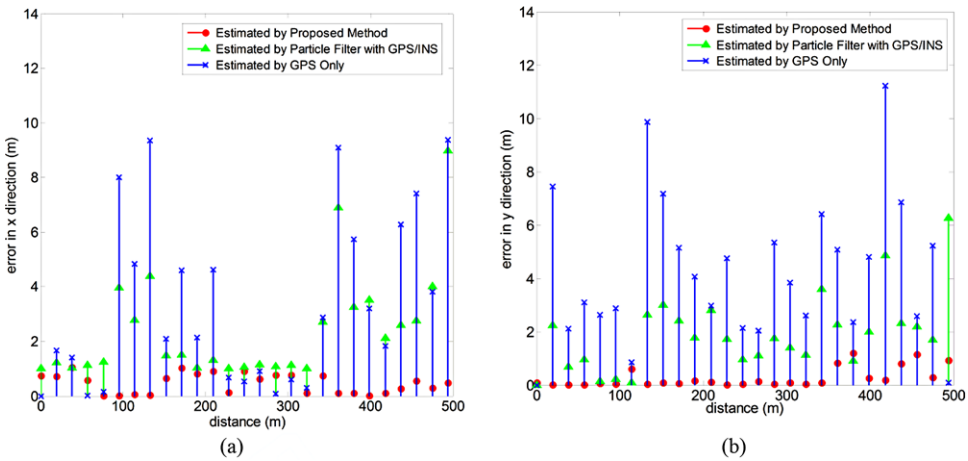

Figure 9. Comparison of position estimation errors in Cartesian coordinates (scenario 1): (a) x direction, and (b) y direction.

very close (0.5 m) to the expected location, while the estimation by GPS-only and the particle filter with GPS/INS was about 8 and 4 m away from the real destination, respectively.

It is found that when the robot moves along a relatively long, straight, and flat roadway that has few terrain variations, the particles may tend to scatter away because of accumulated localization error. This will result in the estimated position falling behind the real position. However, this phenomenon could be eliminated efficiently with a vision-based supplement (Figure 12). According to Figure 12(b), the estimation errors around the turning point E were efficiently decreased with the combination of a vision-aided supplement in the proposed method compared with Figure 12(a).

Image matching performance is evaluated in this scenario as well. For a typical outdoor environment, illumination/shadow conditions varied randomly due to different types of weather. For instance, the left image in Figure 13(a) is the reference image taken on a cloudy day, while the right is the image taken by the robot on a sunny day. Figure 13(b) shows the matching result of two images, where the small circles in different colors are the feature points, and the lines connect the matched feature points. According to Figure 13(b), the matching result is closely related to the building part where most of the reliable matches are generated. The illumination changes caused by different types of weather have a relatively small impact on the variation of the gradient of the buildings in the image. Therefore, the SURF-based matching is not influenced

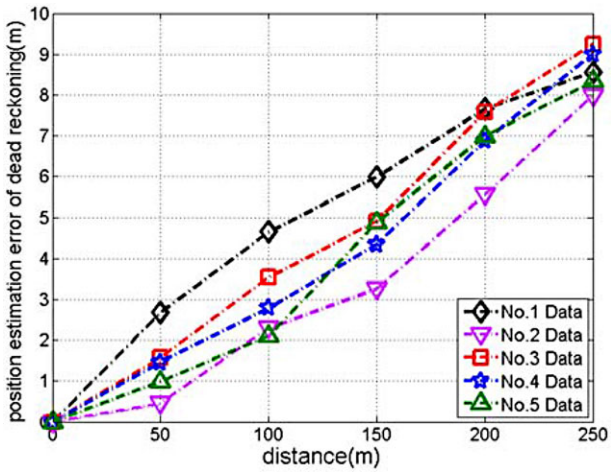

Figure 10. Position estimation errors of dead reckoning with network disconnection (the five lines represent five different records).

under most conditions except when colorful lights shine in some particular angle on the building, causing an obvious variation of the gradient between two images. Figure 14 shows a case of the reference image [Figure 14(a), left] with a stage on the ground and other decorations on the building, while the stage and decorations have been removed in the image taken by the robot [Figure 14(a), right] in real time. Figure 14(b) shows the matching result. Since most of the main features of the building where most of the reliable matches are generated are not changed between two images, this kind of change did not influence the matching result according to Figure 14(b). However, in some uncommon situations when there are some objects (a car parked at the roadside or people around) blocking the main features in the image, this would cause a failure of the matching process. Nevertheless, if the matching fails, then the current reference image will be abandoned and the result will not be sent back to the robot. A small number of failures would not produce any side effects to the localization performance.

The camera mounting position and elevation angle on the robot is not strictly constrained except that it should point to the lateral side of the road in our experiments. Figure 15 shows images taken with different mounting positions and elevation angles. Figure 15(a) is the reference image. Figures 15(b) and 15(c) are taken at a height of 0.5 m with different elevation angles. Figures 15(d) and 15(e) are taken at a height of 1.2 m with different elevation angles. All four images could match with the reference image successfully in the proposed method because the main features of the building would not be changed due to variations of the camera mounting position and elevation angles.

Although most wrong matches could be avoided due to the strict definition of a successful match, in some special situations, such as two identical buildings or objects at different places, a wrong match might still happen. Under these circumstances, the proposed framework has additional strategies to handle the problem. First, only the images within the area of the local road network are searched in the reference image database and matched with the real-time image. Second, if there is a candidate match nearby, the backup IMU and encoder data would help recall the location of the robot where it takes that image and compare it

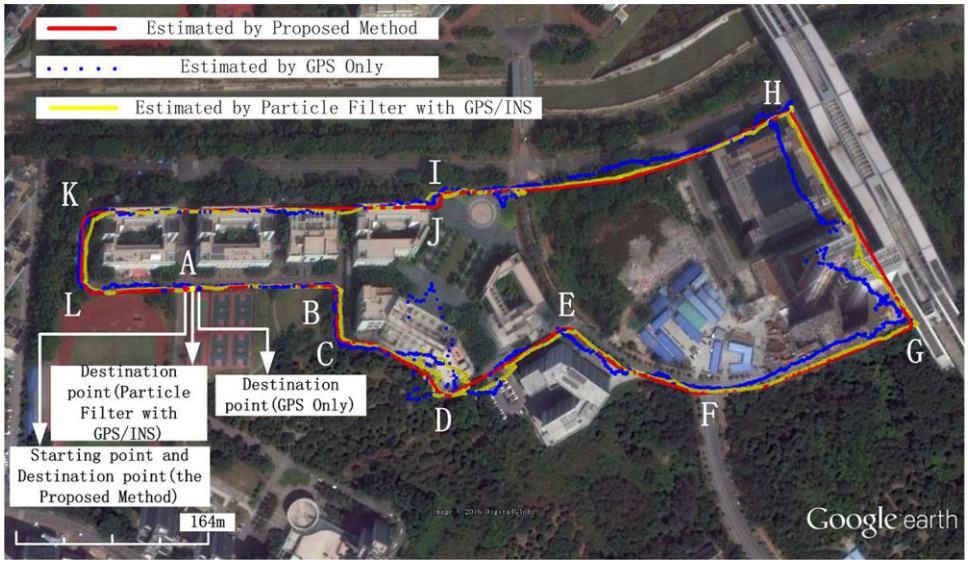

Figure 11. Comparison of position estimation in a large outdoor environment (scenario 2).

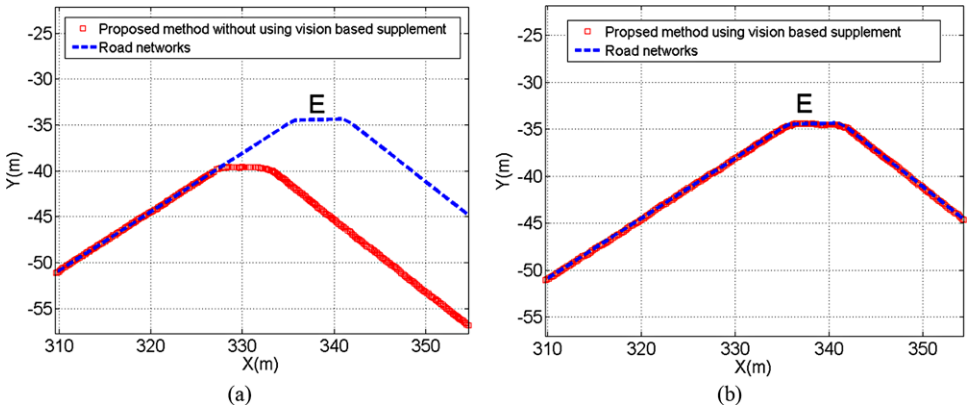

Figure 12. Position estimation (a) without vision-based supplement in turning point *E* in Figure 11, (b) with vision-based supplement.

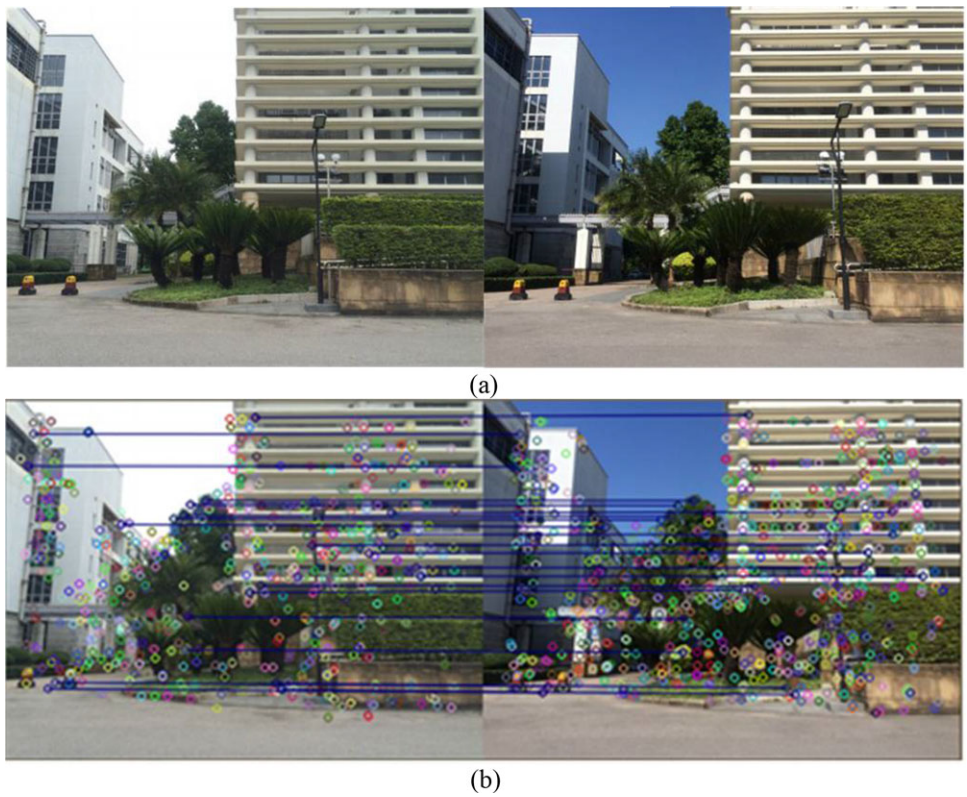

Figure 13. Image matching with changing illumination conditions: (a) reference image taken on a cloudy day (left) and image taken by the robot on a sunny day (right); (b) matching result.

to the location tagged with the reference image. If two locations are much different, a wrong match is detected and the result of the wrong match will not be sent back to the robot. The matching process will restart without any influence on the localization process.

On the other hand, when the robot is moving in an environment with an extremely similar texture, e.g., a road with only trees and other plants by the side, the image matching may fail because there are too few distinguishable features. In such scenarios, sensor resetting could be used to solve the problem. Figure 16 shows a comparison of the localization results with and without sensor resetting when the robot arrived close to the road turning point G in Figure 11. It can be noticed that the error becomes huge

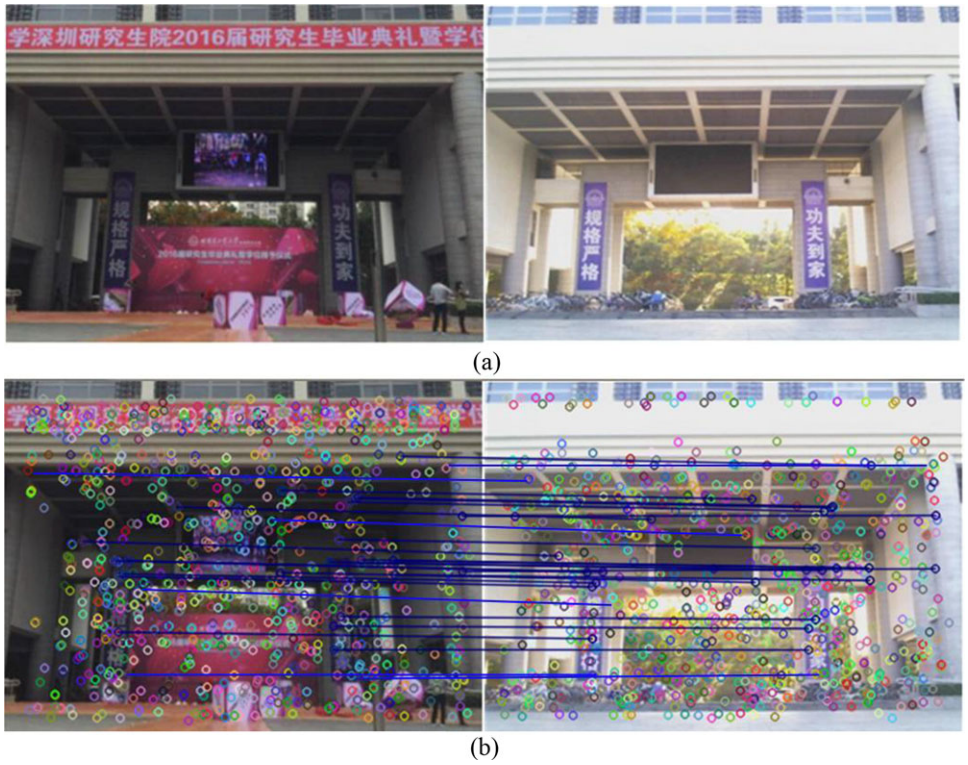

Figure 14. Image matching with changing objects in the images: (a) reference image (left) and image taken by the robot (right); (b) matching result.

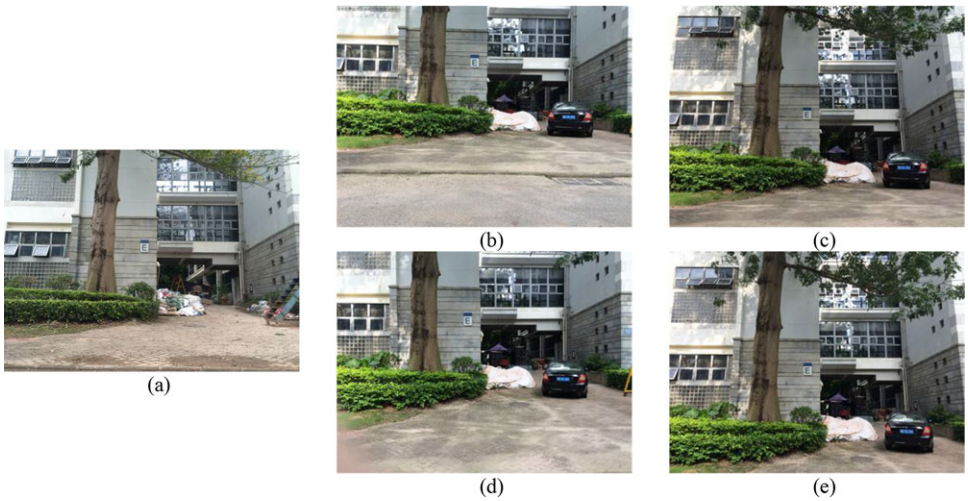

Figure 15. An example of images where (a) is the reference image, and (b)–(e) were taken with different camera mounting positions and elevation angles.

when the robot approaches the turning point because of the image matching failure. However, with the sensor resetting technique, the error could be recovered from failure quickly; see Figure 16(b).

Scenario 3: Figure 17 shows the estimated robot positions by the proposed method in a more general large-scale outdoor environment. The whole distance of this scenario is 13.1 km and covers several areas where GPS signals are

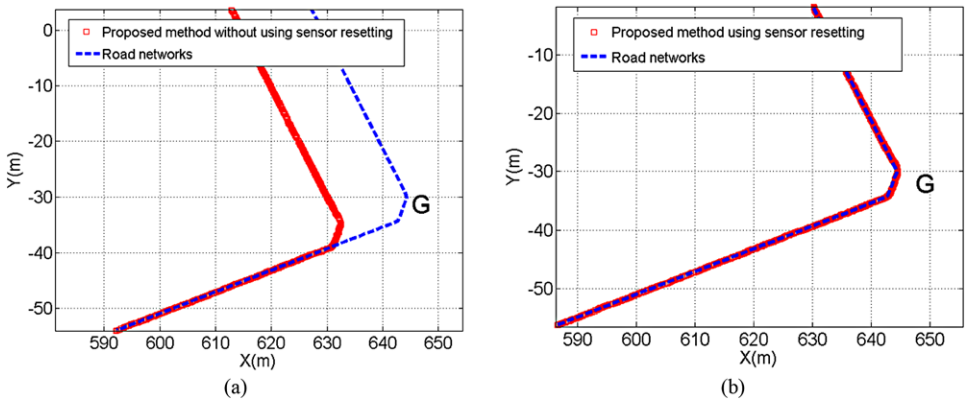

Figure 16. Position estimation on a road with similar texture: (a) without a sensor setting, (b) with a sensor setting.

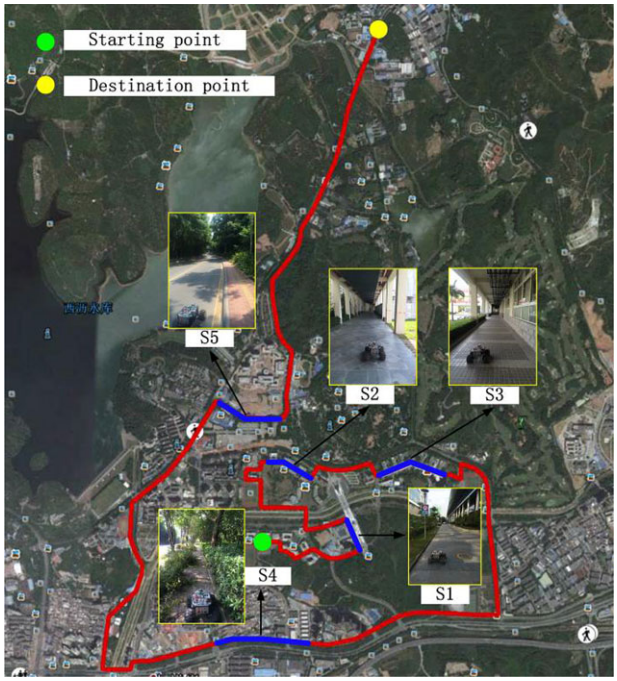

Figure 17. Position estimation in a large-scale outdoor environment.

severely blocked. In Figure 17, the road segment S1 is a long corridor covered by the library with 230 m length that can be treated as the urban canyon because the GPS has deviated far away from the reference on the road. Another two road segments S2 and S3 are long tunnels with 250 and 450 m length where GPS is completely blocked. S4 and S5 are alleyways where the GPS signal is blocked severely by the trees beside the road. Figure 18 shows the estimated position of three segments S1, S3, and S4, where blue triangles represent the estimated position by GPS-only, red asterisks represent for estimated position by particle filter with GPS/INS, black squares represent the estimated position by the proposed method, and pink points represent the reference points. The reference points are the road network points that are extracted in the offline phase. The mobile robot took 3 h and 8 min to finish the whole trip, with an average speed of 1.21 m/s. In addition, no crash of the system happened. The position estimation errors for the whole journey are shown in Figure 19. The error is defined as the Euclidean distance between the estimated point and the reference road network point extracted in the offline phase. The accuracies of those reference points depend on the GIS system, and the error for the reference point was always less than 1 m in our experiments. It is concluded according to Figure 19 that the system performance was quite stable within reasonable estimation errors even in a large-scale complex outdoor environment. It is worth noting that the radius change of the earth would affect localization error when the robot moves in the longitudinal direction over quite a long distance (over 10 or 100 km) because the Cartesian coordinates are used for localization in this paper. In such circumstances, a second initial position would be set to the current location once the robot moves over 10 km longitudinally.

Experimental evaluations in this scenario can also illustrate real-time properties of the proposed cloud framework compared with the traditional way of running all algorithms onboard. According to the area of this scenario (about 2,500 m × 6,700 m), 1 GB storage space is needed for the road network with 1.087 million records and another 4 GB for the reference image database. The computation load could be divided into two main parts, i.e., the RTI model part and the vision-based supplement part. The RTI model part consists of searching for the local road network information and the calculation of the RTI model. It took about 1 ms to search for local road network information on the cloud server (with MySQL in this paper), while it would take the smartphone-level onboard processor (with SQLite) more than 11 s to complete searching if it is running

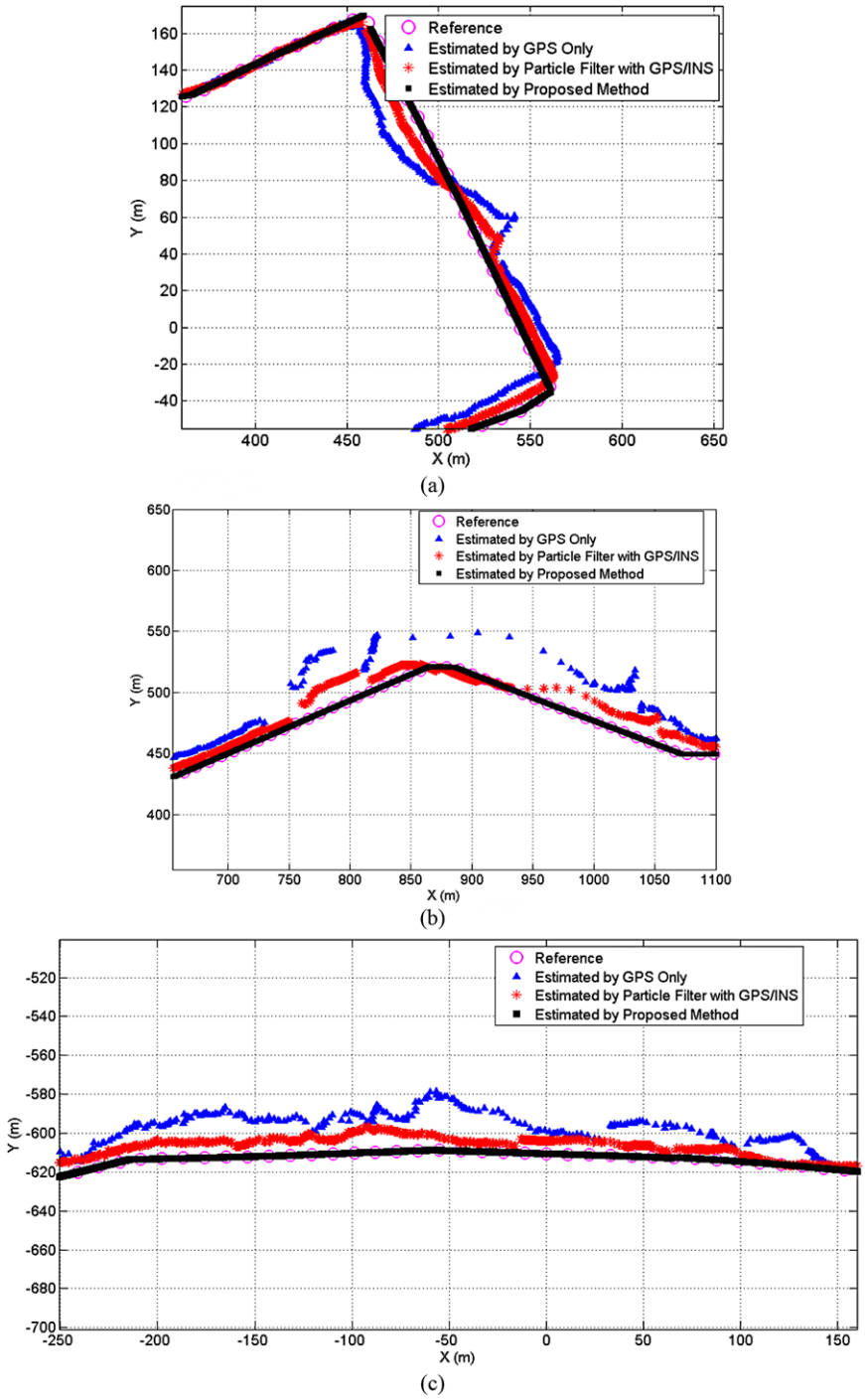

Figure 18. Comparison of position estimation for typical segments: (a) S1, (b) S3, and (c) S4.

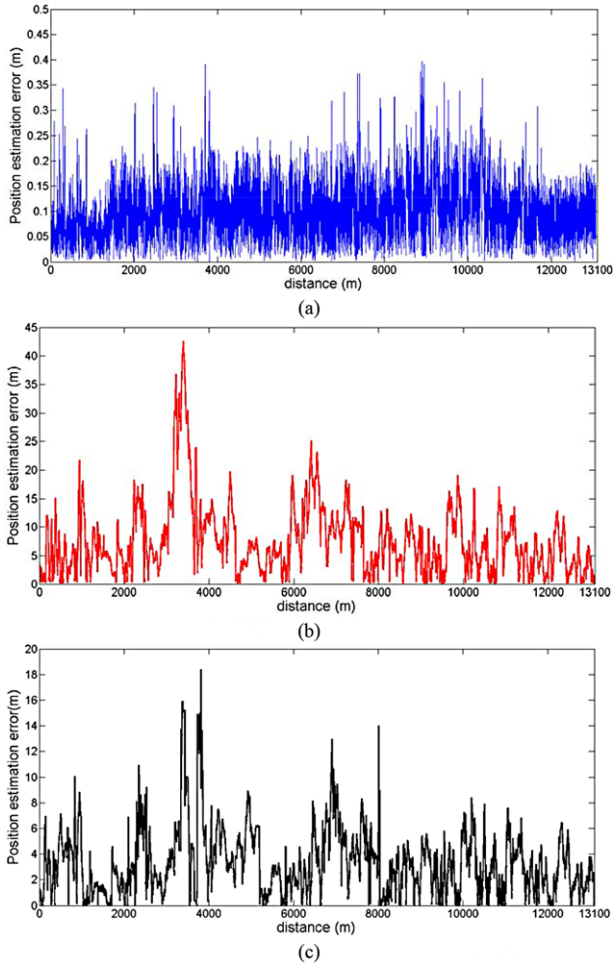

Figure 19. Comparison of position estimation errors: (a) the proposed method, (b) GPS only, and (c) particle filter with GPS/INS.

onboard rather than the cloud server. The calculation of the RTI model on the cloud service took less than 20 ms in this paper, while it would take about 3.5 s for the onboard processor. Similarly, image matching in the vision-based supplement part consumed less than 450 ms and no more than 1.5 s with network delay in the proposed cloud-based framework. However, it would take about 6 s for the low-cost smartphone-level onboard processor. Apparently, in order to meet the requirement for real-time performance, traditional frameworks need onboard processors to be much more demanding and expensive considering the time-consuming issue as traveling areas become larger and larger. Instead, the proposed cloud-based framework has more potential to handle even larger areas (e.g., a city) without greatly leveraging the requirements of the robots.

5. CONCLUSIONS

In this paper, we proposed a cloud-based outsourcing localization architecture for a mobile robot in large-scale outdoor environments. It takes advantage of existing outsourced road network maps and reference images without relying on GPS. Computational and memory loads are mostly distributed on the cloud, while network delay is compensated. The proposed cloud-based framework makes it possible to achieve accurate localization performance without leveraging the requirement of the robot platform as the environment becomes larger and larger. Comprehensive experiments have been conducted to evaluate how different factors affect the performance of the proposed method. Experimental results show that the proposed architecture can be applied to more complex large-scale circumstances. Future work will focus on how to extend the proposed method to autonomous cars where more restrict real-time performance would be demanded and the scale and complexity of the environment are more challenging.

ACKNOWLEDGMENTS

This work was partially presented at the workshop on Cloud Robotics at the 2013 IEEE International Conference on Intelligent Robots and Systems (IROS 2013). This research was supported by the National Science and Technology Ministry of China under Grant No. 2013BAK01B02, Shenzhen Hi-Technology Funding under Grant No. CXZZ20140419141609644, and the National Natural Science Foundation of China under Grant No. 91648102.

REFERENCES

Arumugam, R., Enti, V. R., Liu, B. B., Wu, X. J., Baskaran, K., Kong, F. F., Kumar, A. S., Meng, K. D., & Kit, G. W. (2010). DAvinCi: A cloud computing framework for service robots. In Proceedings of the IEEE International Conference on Robotics and Automation (ICRA), Anchorage, AK.

Axes conventions, Retrieved December 20, 2016, from https://en.wikipedia.org/wiki/Axes_conventions.html.

Barbosa, J. P. D., Lima, F. D. D., Coutinho, L. D., Leite, J. P. R. R., Machado, J. B., Valerio, C. H., & Bastos, G. S. (2015). ROS, Android and cloud robotics: How to make a powerful low cost robot. In Proceedings of the 17th International Conference on Advanced Robotics, Istanbul, Turkey.

Badino, H., Huber, D., & Kanade, T. (2012). Real-time topometric localization. In Proceedings of the IEEE International Conference on Robotics and Automation (ICRA), St. Paul, MN.

Bay, H., Ess, A., Tuytelaars, T., & Van Gool, L. (2008) Speeded-up robust features (SURF). Computer Vision and Image Understanding, 110(3), 346–359.

Blaer, P., & Allen, P. (2002). Topological mobile robot localization using fast vision techniques. In Proceedings of the

IEEE International Conference on Robotics and Automation (ICRA), Washington, DC.

Bonnifait, P., Bouron, P., Crubille, P., & Meizel, D. (2001). Data fusion of four ABS sensors and GPS for an enhanced localization of car-like vehicles. In Proceedings of the IEEE International Conference on Robotics and Automation (ICRA), Seoul.

Bouvet, D., & Garcia, G. (2000). Civil-engineering articulated vehicle localization: Solutions to deal with GPS masking phases. In Proceedings of the IEEE International Conference on Robotics and Automation (ICRA), San Francisco.

Bradley, D. M., Patel, R., Vandapel, N., & Thayer, S. M. (2005). Real-time image-based topological localization in large outdoor environments. In Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems, Edmonton, Alta, Canada.

Brenneke, C., Wulf, O., & Wagner, B. (2003). Using 3D laser range data for slam in outdoor environments. In Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems, Las Vegas.

Chen, H., Li, X., & Ding, W. (2007). Twelve kinds of gridding methods of surfer 8.0 in isoline drawing. Chinese Journal of Engineering Geophysics, 1(4), 52–57.

Drawil, N. M., Amar, H. M., & Basir, O. A. (2013). GPS localization accuracy classification: A context-based approach. IEEE Transactions on Intelligent Transportation Systems, 14(1), 262–273.

Engelson, S. P., & McDermott, D. V. (1992). Error correction in mobile robot map learning. In Proceedings of the IEEE International Conference on Robotics and Automation (ICRA), Nice, France.

Fairfield, N., Kantor, G., & Wettergreen, D. (2007). Real-time SLAM with octree evidence grids for exploration in underwater tunnels. Journal of Field Robotics, 24(1-2), 3–21.

Goldberg, K., & Kehoe, B. (2013). Cloud robotics and automation: A survey of related work (Tech. Rep. UCB-EECS-2013-5). Berkeley, CA: University of California.

Hornung, A., Wurm, K. M., Bennewitz, M., Stachniss, C., & Burgard, W. (2013). OctoMap: An efficient probabilistic 3D mapping framework based on octrees. Autonomous Robots, 34(3), 189–206.

Hu, G. Q., Tay, W. P., & Wen, Y. G. (2012). Cloud robotics: Architecture, challenges and applications. IEEE Network, 26(3), 21–27.

Hygounenc, E., Jung, I. K., Soueres, P., & Lacroix, S. (2004). The autonomous blimp project of LAAS-CNRS: Achievements in flight control and terrain mapping. International Journal of Robotics Research, 23(4-5), 473–511.

Kawewong, A., Tongprasit, N., Tangruamsub, S., & Hasegawa O. (2011). Online and incremental appearance-based SLAM in highly dynamic environments. International Journal of Robotics Research, 30(1), 33–35.

Kehoe, B., Matsukawa, A., Candido, S., Kuffner, J., & Goldberg, K. (2013). Cloud-based robot grasping with the google object recognition engine. In Proceedings of the IEEE International Conference on Robotics and Automation (ICRA), Karlsruhe, Germany.

Kuffner, J. J. (2010). Cloud-enabled robots. In Proceedings of the IEEE-RAS International Conference on Humanoid Robots, Nashville.

Kuipers, B., & Yung-Tai, B. (1991). A robot exploration and mapping strategy based on a semantic hierarchy of spatial representations. Robotics and Autonomous Systems, 8(1-2), 47–63.

Lai, K., & Fox, D. (2010). Object recognition in 3D point clouds using web data and domain adaptation. International Journal of Robotics Research, 29(8), 1019–1037.

Lankenau, A., & Rofer, T. (2002). Mobile robot self-localization in large-scale environments. In Proceedings of the IEEE International Conference on Robotics and Automation (ICRA), Washington, DC.

Lenser, S., & Veloso, M. (2000). Sensor resetting localization for poorly modelled mobile robots. In Proceedings of the IEEE International Conference on Robotics and Automation (ICRA), San Francisco.

Majdik, A. L., Verda, D., Albers-Schoenberg, Y., & Scaramuzza, D. (2015). Air-ground matching: Appearance-based GPS-denied urban localization of micro aerial vehicles. Journal of Field Robotics, 32(7), 1015–1039.

Mandel, C., & Laue, T. (2010). Particle filter-based position estimation in road networks using digital elevation models. In Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems, Taipei, Taiwan.

Martin, M. C., & Moravec, H. P. (1996). Robot evidence grids (Tech. Rep. CMU-RI-TR-96-06). Pittsburgh: Carnegie Mellon University, Robotics Institute.

Miguel, C., & Bonev, B. (2010). Large scale environment partitioning in mobile robotics recognition tasks. Journal of Physical Agents, 4(2), 11–18.

Mohanarajah, G., Usenko, V., Singh, M., D'Andrea, R., & Waibel, M. (2015). Cloud-based collaborative 3D mapping in real-time with low-cost robots. IEEE Transactions on Automation Science and Engineering, 12(2), 423–431.

Nuchter, A., Surmann, H., Lingemann, K., Hertzberg, J., & Thrun, S. (2004). 6D SLAM with an application in autonomous mine mapping. In Proceedings of the International Conference on Robotics and Automation (ICRA), New Orleans.

Pfaff, P., Triebel, R., & Burgard, W. (2007). An efficient extension to elevation maps for outdoor terrain mapping and loop closing. International Journal of Robotics Research, 26(2), 217–230.

Riazuelo, L., Civera, J., & Montiel, J. J. (2014). C(2)TAM: A cloud framework for cooperative tracking and mapping. Robotics and Autonomous Systems, 62(4), 401–413.

Sünderhauf, N., Neubert, P., & Protzel, P. (2013). Predicting the change—A step towards life-long operation in everyday environments. In Proceedings of the RSS Robotics Challenges and Vision Workshop, Berlin.

Thrun, S., Montemerlo, M., Dahlkamp, H., Stavens, D., Aron, A., Diebel, J., Fong, P., Gale, J., Halpenny, M., Hoffmann, G., Lau, K., Oakley, C., Palatucci, M., Pratt, V.,

Stang, P., Strohband, S., Dupont, C., Jendrossek, L., Koelen, C., Markey, C., Rummel, C., Van Niekerk, J., Jensen, E., Alessandrini, P., Bradski, G., Davies, B., Ettinger, S., Kaehler, Adrian., Nefian, A.,& Mahoney, P. (2006) Stanley: The robot that won the DARPA Grand Challenge. Journal of Field Robotics, 23(9), 661–692.

Tongprasit, N., Kawewong, A., & Hasegawa, O. (2011). PIRF-Nav 2: Speeded-up online and incremental appearance-based SLAM in an indoor environment. In Proceedings of the IEEE Workshop on Applications of Computer Vision (WACV), Kona, HI.

Triebel, R., Pfaff, P., & Burgard, W. (2006). Multi-level surface maps for outdoor terrain mapping and loop closing. In Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems, Beijing.

Urmson, C., Anhalt, J., Bagnell, D., Baker, C., Bittner, R., Clark, M. N., Dolan, J., Duggins, D., Galatali, T., Geyer, C., Gittleman, M., Harbaugh, S., Herbert, M., Howard, T. M., Kolski, S., Kelly, A., Likhachev, M., McNaughton, M., Miller, N., Peterson, K., Pilnick, B., Rajkumar, R., Rybski, P., Salesky, B., Seo, Y. W., Singh, S., Snider, J., Stentz, A., Whittaker, W. R., Wolkowicki, Z., Ziglar, J., Bae, H., Brown, T., Demitrish, D., Litkouhi, B., Nickolaou, J., Sadekar, V., Zhang, W., Struble, J., Taylor, M., Darms, M., & Ferguson, D. (2008). Autonomous driving in urban environments: Boss and the Urban Challenge. Journal of Field Robotics, 25(8), 425–466.

Valgren, C., & Lilienthal, A.J. (2007). SIFT, SURF and Seasons: Long-term outdoor localization using local features. In Proceedings of the European Conference on Mobile Robots, Freiburg, Germany.

Valgren, C. & Lilienthal, A. J. (2010). SIFT, SURF and seasons: Appearance-based long-term localization in outdoor environments. Robotics and Autonomous Systems, 58(2), 149–156.

Vandapel, N., Donamukkala, R., & Hebert, M. (2006). Experimental results in using aerial ladar data for mobile robot navigation. In Proceedings of the 4th International Conference on Field and Service Robotics, Mt. Fuji, Japan.

Wang, L.J., Liu, M., & Meng, M.Q.H. (2012). Towards cloud robotic system: A case study of online co-localization for fair resource competence. In Proceedings of the IEEE International Conference on Robotics and Biomimetics, Guangzhou, China.

Xie, J. P., Nashashibi, F., Parent, M., & Favrot, O. G. (2010). A real-time robust global localization for autonomous mobile robots in large environments. In Proceedings of the International Conference on Control, Automation, Robotics and Vision, Singapore.

Xu, D.F., Badino, H., & Huber D. (2014) Topometric localization on a road network. In Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Chicago.

Zhao, H. J., Chiba, M., Shibasaki, R., Shao, X. W., Cui, J. S., & Zha, H. B. (2008). SLAM in a dynamic large outdoor environment using a laser scanner. In Proceedings of the IEEE International Conference on Robotics and Automation (ICRA), Pasadena, CA.

Zhu, X. R., Qiu, C. X., & Minor, M. A. (2013). Terrain inclination aided three-dimensional localization and mapping for an outdoor mobile robot. International Journal of Advanced Robotic Systems, 10(76), 1–9.

Zhu, X. R., Qiu, C. X., & Minor, M. A. (2014). Terrain inclination based three-dimensional localization for mobile robot in outdoor environments. Journal of Field Robotics, 31(3), 477–492.