Engineering Applications of Artificial Intelligence 26 (2013) 1942–1952

Contents lists available at SciVerse ScienceDirect

Engineering Applications of Artificial Intelligence

journal homepage: www.elsevier.com/locate/engappai

Robust omnidirectional mobile robot topological navigation system using omnidirectional vision

Li Maohai [a,b,*], Wang Han [c], Sun Lining [a], Cai Zesu [d]

[a] School of Mechanical and Electric Engineering, Soochow University, Jiangsu, Suzhou 215006, China
[b] School of Mechanical & Aerospace of Engineering, Nanyang Technological University, Singapore 639798, Singapore
[c] School of Electrical & Electronic Engineering, Nanyang Technological University, Singapore 639798, Singapore
[d] School of Computer Science and Technology, Harbin Institute of Technology, Harbin 150001, China

ARTICLE INFO

Article history:
Received 2 June 2012
Received in revised form
11 March 2013
Accepted 18 May 2013
Available online 10 June 2013

Keywords:
Omnidirectional
Mobile robot
Invariant features
Topological map
Navigation
GPU

ABSTRACT

Robust topological navigation strategy for omnidirectional mobile robot using an omnidirectional camera is described. The navigation system is composed of on-line and off-line stages. During the off-line learning stage, the robot performs paths based on motion model about omnidirectional motion structure and records a set of ordered key images from omnidirectional camera. From this sequence a topological map is built based on the probabilistic technique and the loop closure detection algorithm, which can deal with the perceptual aliasing problem in mapping process. Each topological node provides a set of omnidirectional images characterized by geometrical affine and scale invariant keypoints combined with GPU implementation. Given a topological node as a target, the robot navigation mission is a concatenation of topological node subsets. In the on-line navigation stage, the robot hierarchical localizes itself to the most likely node through the robust probability distribution global localization algorithm, and estimates the relative robot pose in topological node with an effective solution to the classical five-point relative pose estimation algorithm. Then the robot is controlled by a vision based control law adapted to omnidirectional cameras to follow the visual path. Experiment results carried out with a real robot in an indoor environment show the performance of the proposed method.

1. Introduction

A key prerequisite for a truly autonomous robot is that it can navigate safely within its environment (Kortenkamp and Bonasso Murphy). The problem of achieving this is the tasks of self-localization, map building, and path planning for an autonomous mobile robot. In this paper three tasks are addressed and new scientific results provided. The difficulty of this problem depends on the characteristics of the robot's environment, the characteristics of its sensors, and the map representation required by the application. The environment could be relatively benign indoors with flat floors. But it could also be quite subversive, at the same time, the size of the environment may also affect the implementation method.

Vision based mobile robot navigation falls into three broad groups: (I) Map based navigation—systems that depend on user created environment maps; (II) Map building based navigation—systems that use sensors to construct their own environment maps and then use these maps for navigation; (III) Mapless navigation—systems that resort to recognizing objects found in the environment or to tracking those objects by generating motions based on visual observations.

Popular choices for the environment maps constructed by robot itself include metrical mapping (Schultz and Adams, 1998), topological mapping (Choset and Nagatani, 2001) and a hybrid approach (Estrada et al., 2005). Metric mapping can be easily build and maintain, however, the problem of large-scale structural ambiguity can be introduced, data association only has to be handled locally and accepting false negative place matches (Bosse et al., 2003). A topological map is a concise description of the large-scale structure of the environment and compactly describes the environment as a collection of places linked by paths. The navigation system proposed in this paper works with natural environments. That means that the environment does not have to be modified for navigation in any way. We choose a topological representation of the environment, rather than a metrical one, because of its resemblance to the intuitive system humans use for navigation, its flexibility, wide usability, memory-efficiency and ease for map building and path planning, Moreover, keeping features in the field of view of an omnidirectional camera is easier than in the case of a perspective camera.

* Corresponding author at: School of Mechanical and Electric Engineering, Soochow University, Jiangsu, Suzhou 215006, China. Tel.: +18862330256.
E-mail addresses: limaohai@163.com, limaohai@suda.edu.cn. (L. Maohai), hw@ntu.edu.sg (W. Han), lnsun@hit.edu.cn (S. Lining), caizesu@hit.edu.cn (C. Zesu).

0952-1976/$ - see front matter © 2013 Elsevier Ltd. All rights reserved.
http://dx.doi.org/10.1016/j.engappai.2013.05.010

For feature extracted from vision systems, in recent years local image feature detectors have bloomed. The most important property of them is their invariance, especially for scale, rotation and affine invariant, such as the Harris point detector (Harris and Stephens, 1988), the Harris–Laplace, Hessian–Laplace, and difference-of-Gaussian (DoG) region detectors (Mikolajczyk and Schmid, 2004; Lowe, 2004; Février, 2007). Some moment-based region detectors (Baumberg, 2000) including the Harris-affine and Hessian-affine region detectors (Mikolajczyk and Schmid, 2004), an edge-based region detector (Tuytelaars and Van Gool, 2004), an intensity-based region detector (Tuytelaars and Van Gool, 2000), an entropy-based region detector (Kadir et al., 2004), and maximally stable extremal region (MSER) (Matas et al., 2004) has been demonstrated to have often better performance than other affine invariant detectors. While all these invariant feature extracting methods need more computation time, fortunately, the graphical processing unit (GPU) has been introduced to meet the demand for invariant features in real-time applications. The feature processing can be accelerated in a parallel computing architecture. Sinha et al. Sinha et al. (2007) and Hedborg et al. Hedborg et al. (2007) demonstrated their real-time GPU-accelerated KLT for more than 1000 features based on a translation model. We implemented the invariant feature extraction and matching based on the SIFT-GPU and KLT-GPU software code ($author1$ et al., changchang; $author1$ et al., Sinha) and affine-SIFT (ASIFT) was proposed by Jean-Michel Morel (Morel and Yu, 2009), and this GPU implementation can achieve a large speedup over CPU.

Topological mapping and localization is a well studied area especially for larger environment (Angeli et al., 2009; Murillo et al., 2007; Guerrero et al., 2008; Goedemé et al., 2007). Nevertheless, there are many differences between the proposed methods used in the omnidirectional vision system to perform topological mapping and localization. For example, Fast Affine invariant features based on GPU are used for the omnidirectional system compared to (Angeli et al., 2009) and radial features in (Murillo et al., 2007; Guerrero et al., 2008), a robust omnidirectional wheel odometry system is used and data from an omnidirectional vision system is used to recover the relative position between views via a least squares estimation technique instead of the 1D trifocal tensor in (Murillo et al., 2007; Guerrero et al., 2008). For parallel GPU computing, which can accelerate the affine invariant feature procession, and need the more flexible and less memory than (Goedemé et al., 2007) using wide baseline features.

The paper is organized as follows: in the following section, the details for the navigation system framework and the omnidirectional motion system are explored, then in Section 3 the robust topological mapping and hierarchical localization algorithm is proposed, and in Section 4 a visual servo algorithm is designed to follow the path. Experiment results and discussions are presented in Section 5 with conclusions in Section 6.

2. System overview

2.1. Navigation system framework

A framework of the navigation system is presented. The system can be divided into three stages: offline map building, online hierarchical localization and autonomous navigation.

The target of the offline map building stage is to build the environment map, which can accurately describe the environment, and the built map can be easily manageable and usable, that is robot navigation and localization can be robustly implemented in this built map. We firstly drive the robot to move through all parts of the environment, and record omnidirectional images and odometry information at a constant time. Later, from this large set of images a topological map is built based on the probabilistic technique and the loop closure detection algorithm. Each topological node provides a set of omnidirectional images characterized by geometrical affine and scale invariant keypoints combined with GPU implementation.

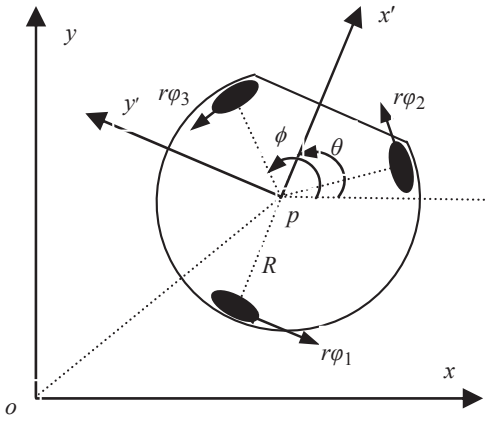


Fig. 2. The layout of three wheels.

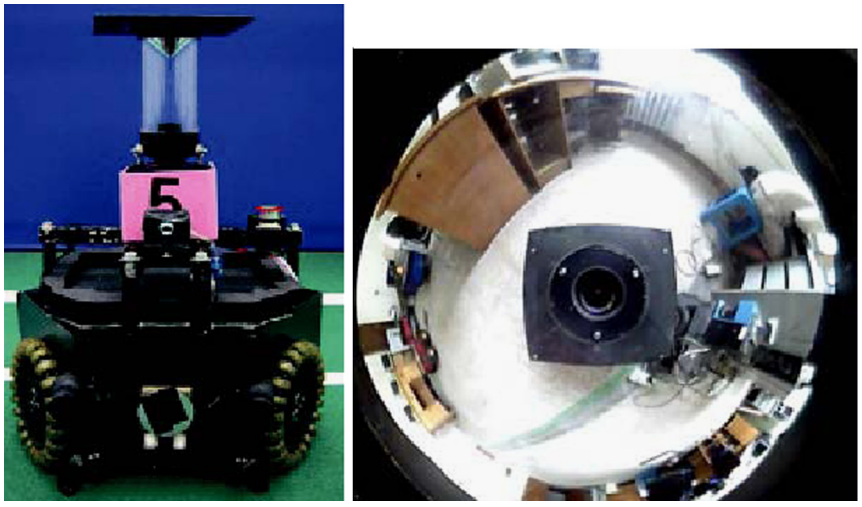


Fig. 1. Omnidirectional mobile robot with omnidirectional vision system.

The next stage is the real-time hierarchical localization. When the system is started up somewhere in the environment, it captures a new image with its omnidirectional camera. This image is compared with the environment map, and a node is firstly located through the node recognition algorithm as fast as possible, and then the more accurate pose of the robot in this node is estimated with the classical five-point relative pose estimation algorithm.

The final stage is path following. When the current node and the relative accurate estimated pose are known, a path can be planned towards the goal. The planned route is specified as a sequence of map images with some keypoints, which are tracked by means of a visual servo algorithm.

2.2. Omnidirectional motion system

As shown in Fig. 1, our mobile robots have an omnidirectional motion system, which is steered by a set of three wheels, and Fig. 2 shows the layout of three wheels. The robot center is represented as p, the robot heading direction is the same as x' axis, the distance R is the robot's radius, r is the wheel's radius, and other parameters are shown in Fig. 2, then the restriction of motion system is shown as follows:

$$J_1R(\theta)\xi + J_2\phi = 0 = \begin{pmatrix} \sin(\varphi/2-\theta)x + \cos(\varphi/2-\theta)y + R\theta + r\phi_1 \\ -\sin(\varphi/2+\theta)x + \cos(\varphi/2+\theta)y + R\theta + r\phi_1 \\ \sin\theta x - \cos\theta y + R\theta + r\phi_1 \end{pmatrix} \quad (1)$$

where

$$J_1 = \begin{pmatrix} \sin(\varphi/2) & \cos(\varphi/2) & R \\ -\sin(\varphi/2) & \cos(\varphi/2) & R \\ 0 & -1 & R \end{pmatrix}, \quad J_2 = \begin{pmatrix} r & 0 & 0 \\ 0 & r & 0 \\ 0 & 0 & r \end{pmatrix}.$$

Because the mobile robot only moves on the flat ground, the Posture Kinematic Model $q = (x\ y\ \theta\ \phi_1\ \phi_2\ \phi_3)^T$is shown as follows:

$$q = \begin{pmatrix} x \\ y \\ \theta \end{pmatrix} = \begin{pmatrix} \cos(\theta) & -\sin(\theta) & 0 \\ \sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} v_1 \\ v_2 \\ \omega \end{pmatrix} \quad (2)$$

$$\begin{pmatrix} \phi_1 \\ \phi_2 \\ \phi_3 \end{pmatrix} = -\frac{1}{r} \begin{pmatrix} \sin(\varphi/2) & \cos(\varphi/2) & R \\ -\sin(\varphi/2) & \cos(\varphi/2) & R \\ 0 & -1 & R \end{pmatrix} \begin{pmatrix} v_1 \\ v_2 \\ \omega \end{pmatrix} \quad (3)$$

Then the above equations allow us to form the inverse Jacobian matrix for the system as follows, which describes the relationship between the velocity of robot itself and the velocity of each wheel.

$$\begin{pmatrix} \phi_1 \\ \phi_2 \\ \phi_3 \end{pmatrix} = -\frac{1}{r} \begin{pmatrix} \sin(\varphi/2-\theta) & \cos(\varphi/2-\theta) & R \\ -\sin(\varphi/2+\theta) & \cos(\varphi/2-\theta) & R \\ \sin\theta & -\cos\theta & R \end{pmatrix} \begin{pmatrix} x \\ y \\ \theta \end{pmatrix} \quad (4)$$

In addition, a dynamic model for the robot can be derived with Newton's Second Law. If m is the weight of the mobile robot, I is the rotation inertia of the mobile robot, and the moment of force for each wheel is τ_1,τ_2,τ_3, the rotation inertia for each wheel is I_φ, then the dynamic model for mobile robot is shown as follows:

$$\begin{pmatrix} mr^2 + 2I_\phi \sin^2(\varphi/2) & 0 & 0 \\ 0 & mr^2 + I_\phi[1 + 2\cos^2(\varphi/2)] & RI_\phi[2\cos(\varphi/2)-1] \\ 0 & RI_\phi[2\cos(\varphi/2)-1] & Rr^2 + 3I_\phi R^2 \end{pmatrix} \begin{pmatrix} \nu 1 \\ \nu 2 \\ \omega \end{pmatrix} = -r \begin{pmatrix} \sin(\varphi/2) & -\sin(\varphi/2) & 0 \\ -\cos(\varphi/2) & \cos(\varphi/2) & -1 \\ R & R & R \end{pmatrix} \begin{pmatrix} \tau_1 \\ \tau_2 \\ \tau_{31} \end{pmatrix} \quad (5)$$

3. Topological mapping and localization

This section describes our method for topological mapping using the omnidirectional vision system. The topological map is built in two steps: first, the initial environment map is obtained in the exploration stage. Given a temporally sampled image sequence acquired during the exploration, the sequence is divided into smaller sections, where each section can be represented as a topological node, and one area might be represented by two or more nodes. In order to obtain a more compact representation of each node, a number of representative image frames are chosen per node area, each characterized by a set of invariant features. Second, the links between the nodes are generated by considering the number of matched feature pairs between the successive frames. The result matrix can show the links. The links typically occur either at places where navigation decisions have to be made or when the appearance of the node changes suddenly.

3.1. Invariant features extraction and matching using GPU

Affine-SIFT is a fully affine invariant image comparison method (Morel and Yu, 2009), which permits to reliably identify features that have undergone very large affine distortions measured by a new parameter, the transition tilt. State-of-the-art methods hardly exceed transition tilts of 2 (SIFT), 2.5 (Harris-Affine and Hessian-Affine) and 10 (MSER). More details are shown in reference (Morel and Yu, 2009). However, this robust method needs more computation time, and the original method is difficult to be used in real-time application. Fortunately, GPU introduced into modern graphics hardware has become increasingly popular for general purpose computations. Especially for computer vision algorithms, whose processing speed can be increased largely compared to their CPU counterparts. So we apply methods that take full advantage of modern graphics card hardware for real-time invariant feature detection and matching.

An implementation of a GPU-KLT/GPU-SIFT algorithm can track more than KLT features in size of 1024×768 with 30 fps (Sinha et al., 2007; Sinha et al., 2006). For realizing our framework, we chose CUDA (NVIDIA) as the programming interface, and we managed to implement the whole procession and avoid any interrupting CPU steps, so the feature list in GPU memory can be transferred easily to the CPU or used on the GPU for further computation. SURF is also a famous GPU implementation using CUDA framework (Cornelis and Gool, 2008). Our implementation runs faster than 30 Hz on 640×480 image frames, which can ensure the real-time omnidirectional path planning. Furthermore, there are some differences from their implementations, besides using different hardware, we make a motion constraint for rotational invariance, yields a somewhat less complex feature extraction and more robust feature matching performance by eliminating of the rotational part of the descriptor.

Because the keypoints are represented as multi-dimensional vector, so we can implement feature matching through finding the feature pairs between current image frame I and set E, which can be implemented with k-nearest search algorithm, and CUDA based GPU computation can speed up the search process.

Given a set E, and a target vector d in E, then a nearest neighbor of d, d' is defined as:

$$\forall d'' \in E, |d \leftrightarrow d'| \le |d \leftrightarrow d''|, \quad |d \leftrightarrow d'| = \sqrt{\sum_{i=1}^{K} (d_i - d'_i)^2} \quad (6)$$

where d_i is the i-th component of d, K is the vector dimension, and Eq. (6) can also be rewritten as matrix computation:

$$|d \leftrightarrow d'|^2 = (d-d')^T(d-d') = \|d\|^2 + \|d'\|^2 - 2d^T d' \quad (7)$$

where $d^T$is the transpose of d, $\|\bullet\|$is the Euclidean norm. Let $P(K \times m)$ and $Q(K \times n)$ are two sets, and containing the m target

keypoints and the n query keypoints, then the distance between these two sets can be written as following:

$$|P \leftrightarrow Q|^2 = N_P + N_Q - 2P^T Q, \quad N_P{}^{(i)} = \|p_i\|^2, \quad N_Q{}^{(j)} = \|q_j\|^2 \tag{8}$$

where $N_P{}^{(i)}$ is the i-th row of N_P, $N_Q{}^{(j)}$ is the j-th column of N_Q. The above equation can be implemented using the CUDA parallel computation which is composed of the following kernels shown in Fig. 3 Garcia et al., 2008; Garcia and Debreuve, 2010.

We do some test on a desktop PC with an Intel Core 2 Quad Core I5 of 2.83 GHz, 4 GB of DDR3 1333 MHz RAM and Nvidia GTX560 1GB. CUDA 3 Beta was used to test. Both CPU and GPU algorithms have the same parameters, and we adapt the five different resolution images. The average result is shown in Fig. 4, and the algorithm performance of GPU implementation is super than CPU implementation.

Although the CPU algorithm implementation can reach near 20 Hz at a small resolution (160*120), but only a little feature points can be extracted as shown in Fig. 4.(2), so this poor quality can cause feature matching to fail, and at the highest resolution 1024*768 the CPU shows very poor performance falling below 1 Hz, while the GPU implementation can reach near 20 Hz, and run faster than 30 Hz on the size of 640×480 image frames, this procession rate is more than adequate for almost all real-time navigation applications. Fig. 4.(3) shows an example of feature extraction, and Fig. 5 shows an example of feature matching, and we set the two sets as $P(K \times m)$ and $Q(K \times n)$, where $n=1024$, $K=64,128$, $m=2^8 \sim 2^{14}$. As shown in Fig. 5, when the number of matched keypoints is smaller, the performance of CUDA computation is not obvious, while the number of keypoints is bigger, the computation can be largely speeded up. Therefore, the speed-up achieved by CUDA implementation in comparison with ANN library (Arya et al., 1998) increased significantly. In our implementation, CUDA was up to 40 times faster than ANN in 128-dimention feature matching.

3.2. Topological mapping

The built topological map constructs environment graph with nodes and links. The nodes are constructed by selecting a set of image frames as similar enough. During the map building, we suppose the traveled distance of mobile robot between each single frame is no great, in this case, when robot is in unstructured environments, we can divide nodes only through local feature matching, and determine the links between nodes, without requiring the geometry information of the environment. On the other hand, when misclassify an image because of noise or occlusion. This can be solved by introducing a probabilistic technique, which deals with the perceptual aliasing problem in indistinctive environments through computing the posterior distribution over the topological space. By inferring the posterior distribution on the whole measurement set, it can be used to confirm the data association between the current observation and the more probable topologies, thus providing the most general solution to the aliasing problem.

While conventional particle filter applies the weighted particle set to approximate the nonlinear and non-Gaussian posterior distribution, if the sampled particles will not lie in the vicinity of real state, the filter diverges easily. In this case, large numbers of particles are needed to approximate the posterior, which can aggravate the computation burden. We apply the evolution operation of genetic algorithm to drive the particles to the posterior density, which can need relatively few particles to approximate accurately the posterior density. For building the same accurate

1. Compute the vector N_P using CUDA (coalesced read / write);
2. Compute the vector N_Q using CUDA (coalesced read / write);
3. Compute the $m \times n$ − matrix $A = -2P^T Q$ using CUBLAS;
4. Add the i^{th} element of N_P to every element of the i^{th} row of the matrix A using CUDA(grid of $m \times n$ threads, non coalesced read/write: use of the shared memory); The resulting matrix is denoted by B;
5. Sort in parallel each column of B (with n threads); The resulting matrix is denoted by C;
6. Add the jth value of N_Q to the first k elements of the j^{th} column of the matrix C using CUDA (coalesced read/write); The resulting matrix is denoted by D;
7. Compute the square root of the first k elements of D to obtain the k smallest distances (coalesced read/write); The resulting matrix is denoted by E;
8. Extract the uppermost $k \times n$-submatrix of E; The resulting matrix is the desired distance matrix for the k-nearest neighbors of each query.

Fig. 3. The nearest search implementation is based on CUDA.

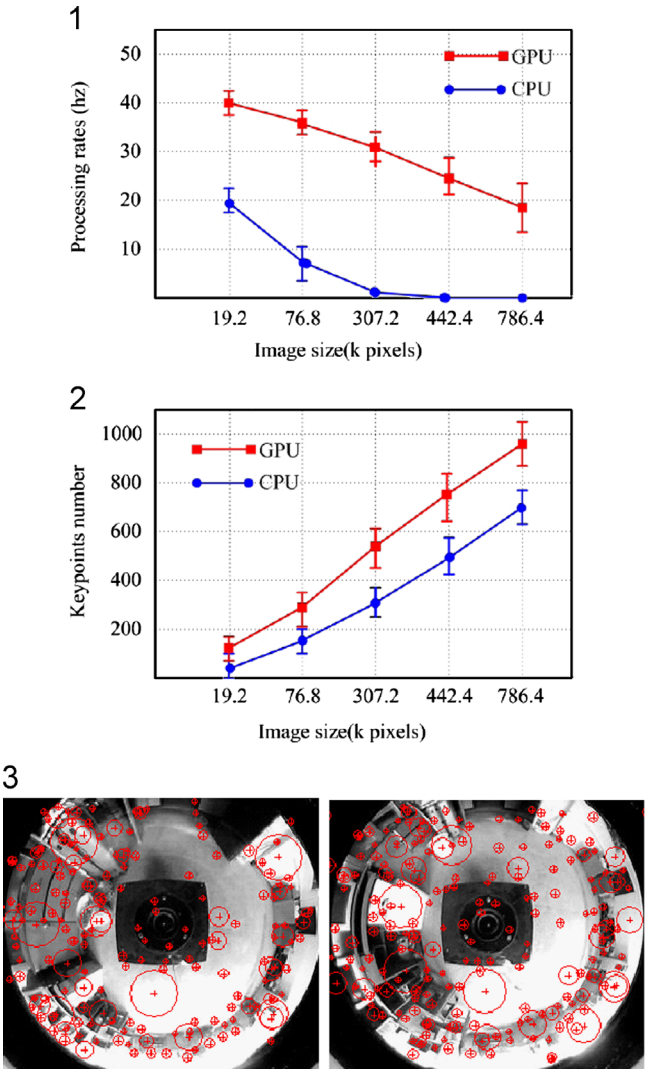
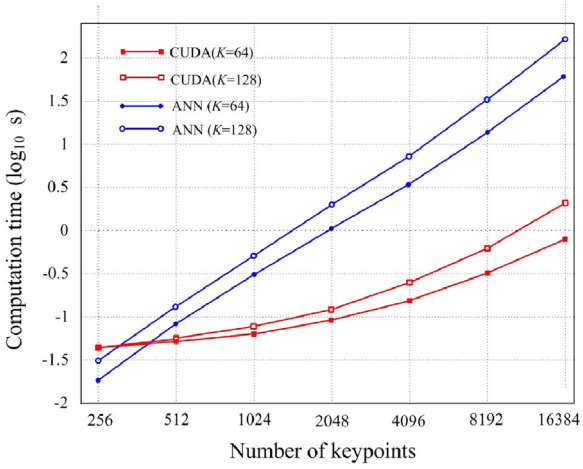
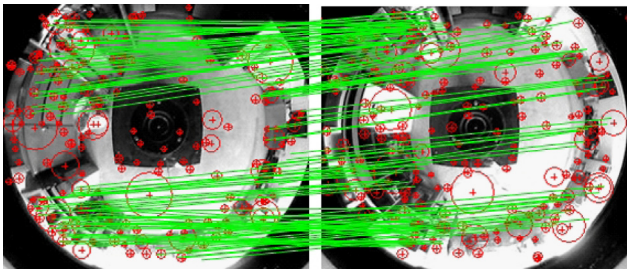


Fig. 4. An example of invariant keypoints extraction. (1) Comparison of processing rate of different image size. (2) Comparison of Keypoints number of different image size. (3) An example.

1



2



3

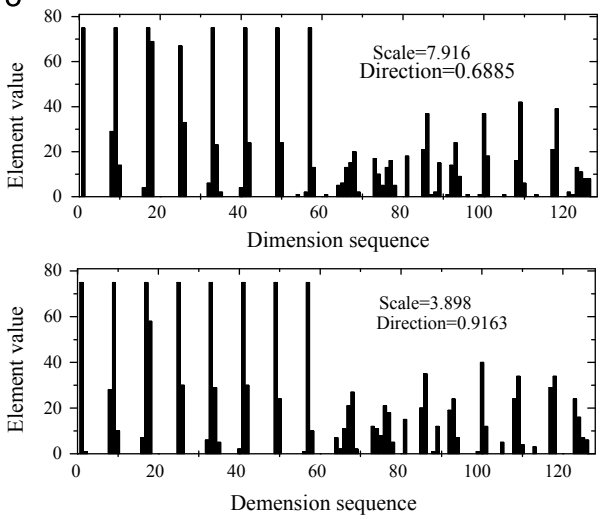


Fig. 5. An example of invariant keypoints matching: (1) Computation time, (2) An example, and (3) the 128-dimenstional vector descriptor for a matching pair.

map, the number of particles needed of proposed method is largely less than the conventional particle filter, which can further reduce the computation burden.

The Generic Particle Filter algorithm is described as following:

(1) Initialization: Sample $S_0=\{(x_0^{(j)},w_0^{(j)})|j=1,\ldots,N\}$ from $p(x_0)$, let $t=1$, $G=5$.
(2) Selection step: Resample N random samples from S_{t-1}, which form the new sample set $\overline{S}_{t-1}=\{(\overline{x}_{t-1}^{(j)},1/N)|j=1,\ldots,N\}$.
(3) Importance sampling step:
   (a) For $j=1,\ldots,N$, sample $x_t^{(j)}$ from $p(x_t|\overline{x}_{t-1}^{(j)},u_{t-1})$ by $\overline{x}_{t-1}^{(j)}$and u_{t-1};
   (b) For $j=1,\ldots,N$, evaluate the importance factor,

$$w_t^{(j)}=\frac{p(y_t|x_t^{(j)})p(x_t^{(j)}|\overline{x}_{t-1}^{(j)},u_{t-1})p(\overline{x}_{t-1}^{(j)}|Y^{t-1})}{p(x_t^{(j)}|\overline{x}_{t-1}^{(j)},u_{t-1})p(\overline{x}_{t-1}^{(j)}|Y^{t-1})}=p(y_t|x_t^{(j)}) \tag{9}$$

(4) Evolution step:
   (a) Initialize: $K=1$;
   (b) Crossover: for $j=1,\ldots,N$. Draw two samples $x_t^{(A)},x_t^{(B)}$from S_t randomly. Then select two samples $x_t^{(Ac)},x_t^{(Bc)}$ with larger factors to replace $x_t^{(A)},x_t^{(B)}$.

$$\begin{cases} x_t^{(Ac)}=\xi x_t^{(A)}+(1-\xi)x_t^{(B)}, \\ x_t^{(Bc)}=\xi x_t^{(B)}+(1-\xi)x_t^{(A)}, \\ w_t^{(Ac)}=p(y_t|x_t^{(Ac)}), \\ w_t^{(Bc)}=p(y_t|x_t^{(Bc)}). \end{cases} \tag{10}$$

   where $\xi\sim U[0,1]$, $U[0,1]$ represents uniform distribution. Mutation: for $j=1,\ldots,N$. Draw a sample $x_t^{(A)}$ from S_t, if $w_t^{(A)}<M_{th}$, then reject it and randomly draw another sample from the state space to replace it, and $x_t^{(A)}$ is mutated. Then select the sample $x_t^{(Am)}$ with larger factor to replace $x_t^{(A)}$.

$$\begin{cases} x_t^{(Am)}=x_t^{(A)}+\tau \\ w_t^{(Am)}=p(y_t|x_t^{(Am)}) \end{cases} \tag{11}$$

   where$\tau\sim N(0,\Sigma)$is a three dimensional vector.
   (c) $K=K+1$; if $K>G$ then go to step (b)
(5) Factor normalization: for $j=1,\ldots,N$, $w_t^{(j)}=w_t^{(j)}/\sum_{k=1}^{N}w_t^{(k)}$.
(6) Output step: Output a set of samples $S_t=\{(x_t^{(j)},w_t^{(j)})|j=1,\cdots,N\}$ that can be used to approximate the posterior density.
(7) $t=t+1$, return to step (2).

For comparing the proposed method with traditional algorithm, we build the map for the same room, as shown in Fig. 6, and (0,0) indicates the initial robot position, the dashed line indicates the estimated path. The arrow indicates robot's movement. Figs. 7 and 8 shows the convergence of the proposed method can be faster, and the traditional algorithm need a large number of particles to gain a high positioning accuracy, and the positioning accuracy is easily influenced by the number of particles. But the algorithm used in this article can reach a higher accuracy by taking only a few particles. When the number of particles reaches a certain number, the positioning accuracy will not be influenced by the number of particles.

3.2.1. Loop closure constraint

Loop closure has gained popularity in recent times. Many loop closure algorithms have been proposed and tested (Zivkovic et al., 2007; Koseck et al., 2005; Tapus and Siegwart, 2005; Christoffer Valgren, 2007; Navid, 2010; Angeli et al., 2008). Most of the previous works were only focused on convex spaces and need larger computation time, while we use the GPU parallel computation without worry about the computation burden. Furthermore we combine the odometry information to the node recognition algorithm, and recursively compute the most likely node with the Bayes probability technique. The final experiments show the superior performance even if under the larger perceptual aliasing conditions.

We define the similarity between the current image frame and the topological map node. Let the i-th node N_i in the model be represented by n views $V_1^{(i)},V_2^{(i)},\ldots,V_n^{(i)}$ with $i=1,\ldots,N$ and each view is represented by a set of keypoints $\{F_k(V_j^{(i)})\}$, where k is the number of features. In the on-line recognition stage, features $\{F_k(Q)\}$ detected from the input image Q are pushed into the search engine to find the most similar match $\{M(Q,N_i)\}$. The similar

node of the query image Q is then determined based on the number of successfully matches:

$$Q[n, num] = \max_i \{M(Q, N_i)\} \tag{12}$$

where n is the index of node with maximum number num of matched keypoints.

The goal of the loop closure detection is to compute a probability distribution over the possible nodes given all the extracted keypoints and robot movements up to time t. Let a set $S_t=\{s_0,\ldots,s_t\}$ denotes the node set, which represents the loop closure hypotheses at time t, and let $z_{1:t}$ and $o_{1:t}$ denote as the features set and the robot control inputs up to time t respectively, and$o_t$means the robot pose variation $(\Delta d_t, \Delta\theta_t)$ from time t−1 to t, Δd and $\Delta\theta$ represents respectively as the translation and heading angle changes. Then

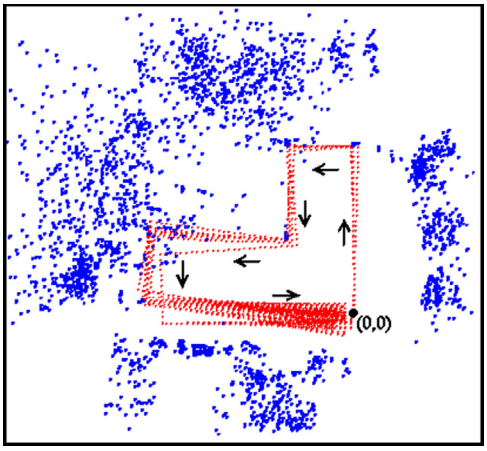


Fig. 6. Bird's-eye view of the landmarks in the map.

$P(S_t|z_{1:t}, o_{1:t})$ can be computed recursively as following:

$$\begin{aligned} Bel(S_t) &= P(S_t|z_{1:t}, o_{1:t}) = \eta P(z_t|S_t, o_{1:t}) P(S_t|z_{1:t-1}, o_{1:t}) \\ &= \eta P(z_t|S_t) P(S_t|z_{1:t-1}, o_{1:t}) \quad \text{Markov} \\ &= \eta P(z_t|S_t) \int P(S_t|o_t, S_{t-1}) Bel(S_{t-1}) dS_{t-1} \end{aligned} \tag{13}$$

where η is the normalization term, and $P(S_t|o_t, S_{t-1})$ is the transition matrix and $P(z_t|S_t)$ is the observation density. For the computation convenience, we approximate$P(z_t|S_t)$ with a mixture of K Gaussians models:

$$p(z_t|S_t) = \sum_{k=1}^{K} P(\xi_t = k|S_t) p(z_t|S_t, \xi_t = k) \propto \sum_k W(S_t, k) \exp\left(\frac{-1}{2\sigma_s^2} \|z_t - \mu_{k,s}^s\|^2\right) \tag{14}$$

where ξ_t is the latent indicator variable, and $W(S_t, k)$ is the weight of mixture component k givenS_t.

Once we acquire the m highest loop closure hypotheses, then we further check the validation of each of hypotheses, and select the hypothesis of the smallest 2D motion variation as the final result. The core process for the loop closure algorithm is shown in Fig. 9. Fig. 10 shows some loop closing experiment results.

3.3. Hierarchical topological and metric localization

We proposed the hierarchical topological and metric localization method, and the topological localization can ensure the coarse global localization as fast as possible, while the metric localization can ensure the accuracy of the relative pose in topological node. The proposed loop closure detection algorithm is suitable to the global localization. Once the topological node lied in is confirmed, we further use the classical five-point relative pose estimation to accurately localize the robot (Nistér, 2004).

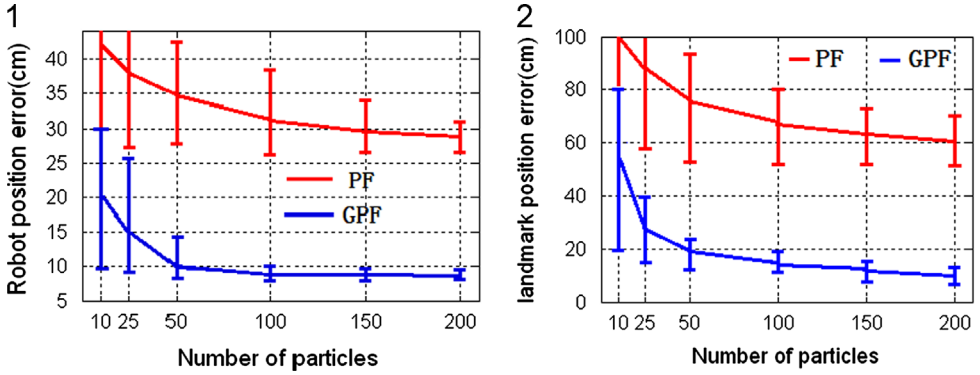


Fig. 7. The positioning accuracy influenced by the number of particles: (1) robot position estimation error, (2) landmark estimation error.

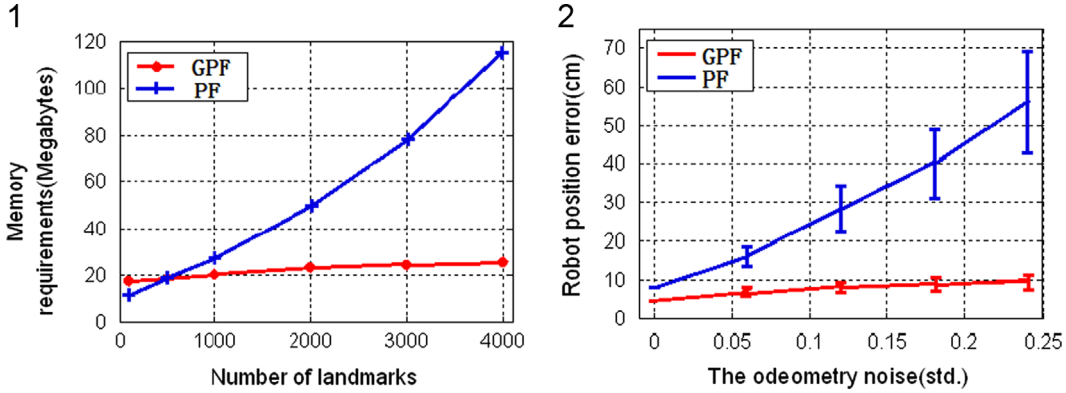


Fig. 8. Performance comparison of different number of landmarks and odometry noise: (1) memory requirements, (2) robot position error.

Input : Current node set S_t;
Define : *Extracting*( ) - function that extracts features
Matching( ) - function that returns similartiy
$Vector_h$( ) - vector that stores the m highest hypotheses
Ransac() - function that returns the 2D motion
$Vector_d$( ) - vector that stores *Ransac*() results
Parameters : Threshold for translation change Δd_T, heading change $\Delta\theta_T$, 2D distance variation D_T, loop closure hypotheses P_T, matching pairs M_T
1 : Capture current image I_t and Odometry $o_t(\Delta d_t, \Delta\theta_t)$;
if $\Delta d_t < \Delta d_T$ or $\Delta\theta_t < \Delta\theta_T$ Continue;
2 : Invariant features $z_t := Extracting(I_t)$;
3 : *bool Similar* := $Matching(z_t, z_{t-1})$
4 : if *Similar* := True
go to step 1.
else Compute $P_{st} := P(S_t | z_{1:t}, o_{1:t})$ with *equation* (18);
5 : Push the higher hypotheses $P_{st} > P_T$ to $Vector_h$( );
end if
if $Vector_h$( ) := Null
6 : add a new node to the map;
else for all highest hypotheses in $Vector_h$( )
if feature pairs smaller than M_T Continue;
else compute 2D motion with *Ransac*( );
7 : Push the *Ransac*( ) < D_T to $Vector_d$( );
end if
end for
if $Vector_d$( ) := Null
8 : add a new node to the map;
else select S' with the smallest from $Vector_d$( );
9 : Associate I_t with S', and correct map with the relaxation algorithm;
end if
end if
10 : *go to* step 1.

Fig. 9. The loop closure in mapping process.

The metric localization experiments are executed in three different rooms, and at five different positions in each room. Robot localizes itself for 10 times at each position, as shown in Fig. 11, simultaneously, we analyze these final localization errors. The results are shown in Fig. 11b and c, the localization error of translation can be smaller than 10 cm, which can be adequate for the navigation requirements.

4. Visual-servo for navigation

The robot state consists of a vector $S=(x, y, \theta, v, \omega, t)=(m, t)\in R^6$, where $(u_{xt}, u_{yt})\in R^2$, $(v_{xt}, v_{yt})\in R^2$, $(x_t, y_t)\in R^2$, and $\theta_t\in[0, 2\pi)$ denote the control input, velocity, position, and heading of the robot respectively, at the time t. The navigation system can modify the robot's linear and angular velocities denoted by (v, ω). The robot dynamic model is:

$$\begin{aligned} &\dot{x}=v\cos\theta, \quad \dot{y}=v\sin\theta, \quad \dot{\theta}=\varpi \\ &\dot{x}\cdot\sin\theta-\dot{y}\cdot\cos\theta=0 \end{aligned} \tag{15}$$

The tracking path P is defined as a set of state points $S_P, p\in\{0, 1, 2\ldots\}$. At each time instant, the path tracking module must determine a target point on the trajectory after time interrupt T, $S'=(x', y', \theta', v', \omega', t')$ which is then used to determine the target position and orientation to initial state S_0, then we can obtain the following equation:

$$\begin{aligned} &t'=t_0+T \\ &v_x{}'=v_{x0}+u_x\cdot T, \quad v_y{}'=v_{y0}+u_y\cdot T \\ &\omega'=\tfrac{v_y-v_x}{L}=\tfrac{v_{y0}-v_{x0}}{L}+\tfrac{T}{L}(u_y-u_x)=\omega_0+\tfrac{T}{L}(u_y-u_x) \\ &v'=\tfrac{v_y+v_x}{2}=\tfrac{v_{y0}+v_{x0}}{2}+\tfrac{T}{2}(u_y+u_x)=v_0+\tfrac{T}{2}(u_y+u_x) \\ &\theta'=\theta_0+\textstyle\int_0^{0+T}\omega(t)dt=\theta_0+\omega_0 T+\tfrac{T^2}{2L}(u_y-u_x) \end{aligned} \tag{16}$$

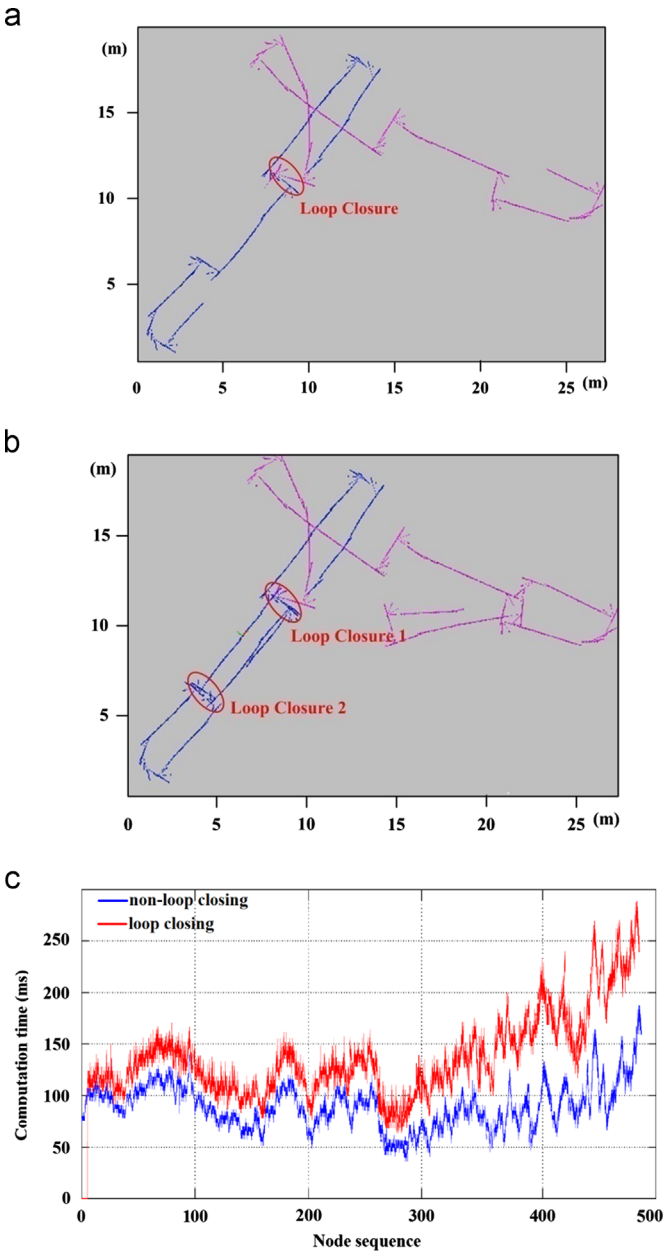


Fig. 10. Example of the loop closure: (a) one loop closing, (b) two loop closing, and (c) Comparison results of the computation time whether the loop closing is combined in mapping or not.

For obtaining the differential results through computing the value of x', y', we think about the following simplified equation:

$$\begin{aligned} &\cos(\theta')=\cos\left(\theta_0+\omega_0 t+\tfrac{t^2}{2L}(u_y-u_x)\right)\approx\cos(\theta_0+\omega_0 t) \\ &\sin(\theta')=\cos\left(\theta_0+\omega_0 t+\tfrac{t^2}{2L}(u_y-u_x)\right)\approx\sin(\theta_0+\omega_0 t) \end{aligned} \tag{17}$$

Simultaneously, according to the following basic formula of undefined integral:

$$\begin{aligned} &\textstyle\int x\cos x dx=\cos x+x\sin x+c, \\ &\textstyle\int x\sin x dx=\sin x-x\cos x+c \end{aligned} \tag{18}$$

We can obtain the following equations:

$$x'=x_0+\int_{t_0}^{t_0+T} v\cdot\cos\theta\cdot dt$$

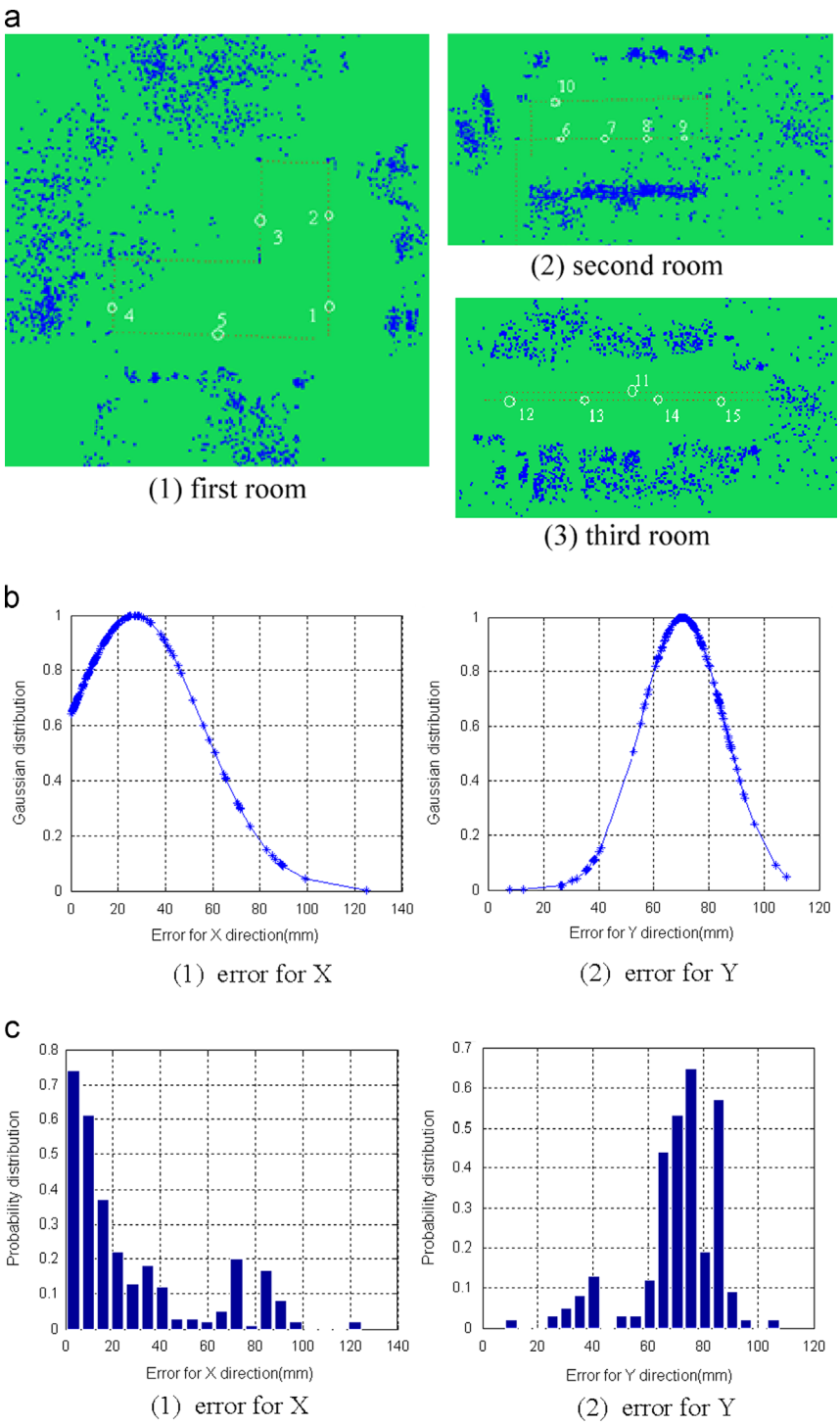


Fig. 11. An example of the metric localization: (a) the metric localization experiments in three rooms. (b) error Gaussian distribution: (1) error for X, (2) error for Y. (c) error histogram: (1) error for X, (2) error for Y.

$$\begin{aligned} &\approx x_0 + \int_{t_0}^{t_0+T} \left[v_0 + t/2(u_y + u_x)\right] \cdot \cos(\theta_{mid}) \cdot dt \\ &= x_0 + v_0 \cdot \cos(\theta_{mid}) \cdot T + \frac{T^2}{4}(u_y + u_x) \cdot \cos(\theta_{mid}) \\ &\quad + \frac{t_0 \cdot T}{2}(u_y + u_x) \cdot \cos(\theta_{mid}) \end{aligned} \tag{19}$$

$$y' = y_0 + \int_{t_0}^{t_0+T} v \cdot \sin\ \theta \cdot dt$$

$$\begin{aligned} &\approx y_0 + \int_{t_0}^{t_0+T} \left[v_0 + t/2(u_y + u_x)\right] \cdot \sin(\theta_{mid}) \cdot dt \\ &= y_0 + v_0 \cdot \sin(\theta_{mid}) \cdot T + \frac{T^2}{4}(u_y + u_x) \cdot \sin(\theta_{mid}) \\ &\quad + \frac{t_0 \cdot T}{2}(u_y + u_x) \cdot \sin(\theta_{mid}) \end{aligned} \tag{20}$$

From the above equations, we can easily obtain the variant value of x', y', θ', v', ω', they all relate the mobile robot control variants u_x, u_y, through these inputs, the mobile robot motion state

Let s = (m,t), $\tau :| s \to t$, $\tau(s) = t$, $U = \{u_1, u_2, ..., u_n\}$
Initial: $L = H = \{s_b\}$, $E(s_g) \neq \varnothing$;
if $(s_b \in E(s_g))$ *return* (*SUCCESS*);
else {
  while$(H != \varnothing)$ {
    Get the first element s_0 from H;
    if $(\tau(s_0 \geq t_b + M * T))$
      {*return* (*FAILURE*); *goto* *END*;}
    else {
      for$(i = 1, i \leq n, i++)$ {
        $m_i = m_0 + \int_{t_0}^{t_0+T} f(m,u,t); t_i = t_0 + T;$
        $s_i = s(s_0, u_i) = (m_i, t_i);$
        //Calculate s_i from s_0 under u_i through time T
        if $(s_i \in E(s_g))$ {
          Get $u : \lfloor t_b\ t_g \rfloor \to U$ using Backtracking;
          return (*SUCCESS*); *goto* *END*; }
        else {
          if $(s_i \in E(s_g))$ {
            $L \leftarrow L \cup \{s_i\}$; $H \leftarrow H \cup \{s_i\}$;
            //Insert state s_i into Queue end of H, L }
          }
        }//for continue
      $H \leftarrow H \cup \{s_0\}$;
    }//while continue
    return (*FAILURE*);
  }//end else
END

Fig. 12. Path planning algorithm.

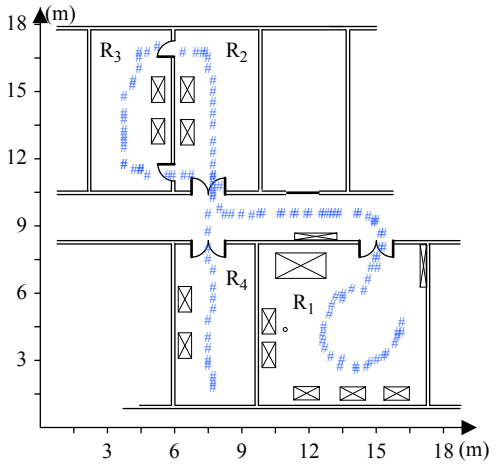


Fig. 13. The environment model and robot's path.

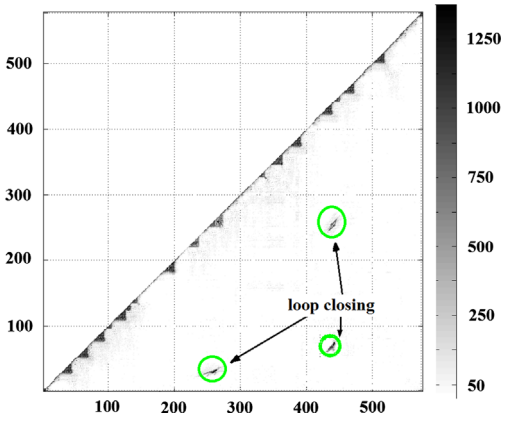


Fig. 14. Feature matches matrix.

can be transferred any state from initial state s_0. For arriving at the target state s_g, we build a search tree, and the root for tree is initial state s_0. When the search arrives at the near-neighbor $E(s_g)$, the search will stop, then we can obtain the final path and control region: $u : \lfloor t_b\ t_g \rfloor \to U$through backtracking method. More details for path planning algorithm are shown in Fig. 12.

5. Experimental results

The online navigation experiments are performed on a mobile robot system incorporating a 3.0 GHz Intel Pentium processor as shown in Fig. 1, which also provides motor control and the main power for vision processing and the navigation software. An omnidirectional color CCD camera mounted at the top of the robot is used to extract the keypoints. The test environment is composed of four rooms R_1, R_2, R_3, R_4 and a horizontal corridor, which is shown as Fig. 13.

For illustrating the robustness of our topological navigation method with an omnidirectional camera, we firstly drive the robot to capture and record the image frames, from this sequence the topological map of this environment is built. First, we capture a database of 826 color images with a size of 640 × 480 pixels. The total traveled path length was approximately 63 m. We implement the topological mapping on a desktop PC with an Intel Core 2 Quad Core I5 of 2.83 GHz, 4 GB of DDR3 1333 MHz RAM and Nvidia GTX560 1GB, which has the powerful image processing ability with CUDA parallel GPU implementation, so the recorded odometry information and ominidirectional image frames are combined to generate the robust and accurate topological map with the proposed methods.

After building topological map, only about 578 image frames are reserved, other images are omitted for false matches and false links. The final feature matches matrix between these images is shown in Fig. 14. In Fig. 15, an example of the resulting topological map can be seen, the circle represents the key node in the vision path, and the links between the nodes indicate the relationship, darker lines indicate links with much more feature matches. It is interesting to note that there is also false links and loop closing. These links exist because of perceptual aliasing and traveled path overlapping. We use this topological map for the final navigation experiments. The initial robot poses are located near the planning path with the orientation similar to the target paths. As a result, in most cases, the robot pose successfully converges to the target node, while the failed cases are caused by the false localization results. We have tested the node converging performance of the control rule as shown in Fig. 16, which shows the error changes of robot state with the control input. Fig. 17 shows an example of iterative converging process.

6. Conclusion

This article has proposed a topological autonomous navigation system for an indoor mobile robot with an omnidirectional camera. The omnidirectional navigation system is composed of on-line and off-line stages. During the off-line learning stage, the robot performs paths and records a set of ordered key images. Given a topological node as a target, the robot navigation mission

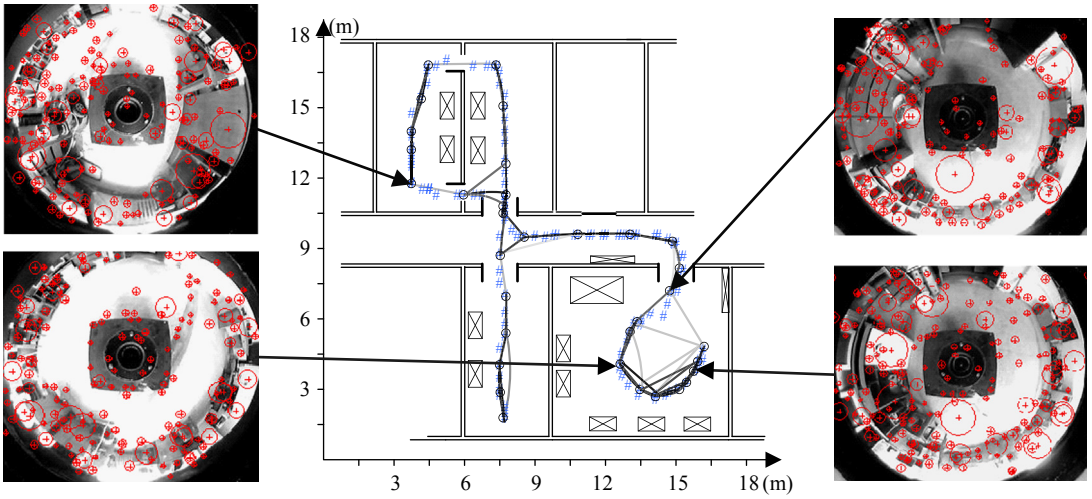


Fig. 15. A topological map for test environment.

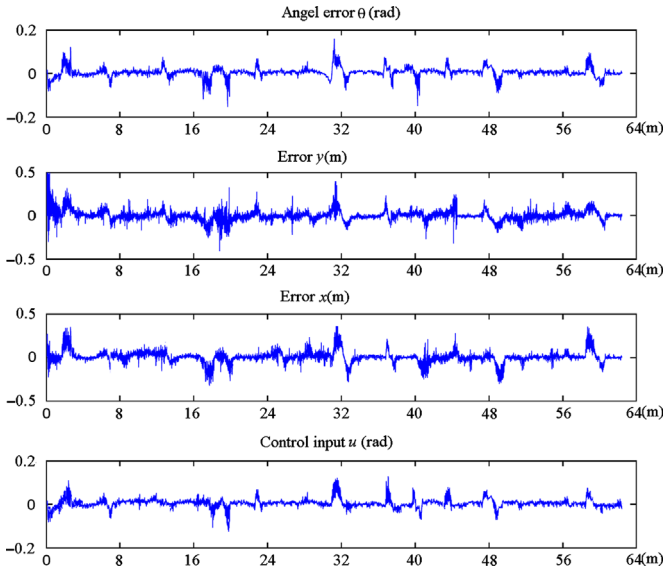


Fig. 16. Error changes of robot state with the control input.

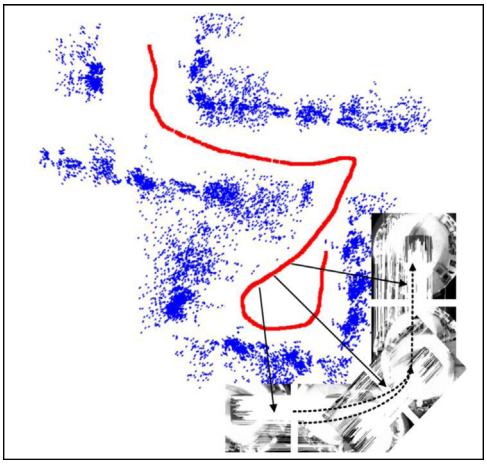


Fig. 17. Examples of iterative converging process.

is a concatenation of topological node subsets. In the on-line navigation stage, the robot recognizes the most likely node through robust node recognition algorithm, and estimate the accurate metric pose in this node, then controlled by a vision based control law adapted to follow the visual path. Experiment results show the performance of the proposed method.

References

A. Angeli, D. Filliat, S. Doncieux, and J.-A. Meyer, A fast and incremental method for loop-closure detection using bags of visual words, IEEE Trans. Robotics, 24, (October (5)), 1027–1037, 2008 (special issue on Visual SLAM).

Angeli, A., Doncieux, S., Meyer, J.-A. and Filliat, D., Visual topological SLAM and global localization. In: IEEE International Conference on Robotics and Automation, 2009.

Arya, S., Mount, D.M., Netanyahu, N.S., Silverman, R., Wu, A.Y., 1998. An optimal algorithm for approximate nearest neighbor searching fixed dimensions. J. ACM 45, 891–923.

Baumberg, A., Reliable feature matching across widely separated views. In: Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 1, 2000, 774–781.

Bosse, M., Newman, P., & Leonard, J., et al. An Atlas framework for scalable mapping. Proceedings of the IEEE International Conference Robotics and Automation. Taiwan, 2003, 1899–1906.

Choset, H., Nagatani, K., 2001. Topological simultaneous localization and mapping (SLAM): toward exact localization without explicit localization. IEEE Trans. Robotics Autom. 17 (2), 125–137.

Christoffer Valgren, Tom Duckett, and Achim Lilienthal. Incremental spectral clustering and its application to topological mapping. In: Proceedings of the IEEE International Conference on Robotics and Automation, 4283–4288, 2007.

Cornelis, N. and Gool, L. V.. Fast scale invariant feature detection and matching on programmable graphics hardware. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2008.

Estrada, C., Neira, J., Tardos, J.D.. Hierarchical SLAM: real-time accurate mapping of large environments. In: IEEE International Conference on Robotics and Automation, 2005, 588–596.

Février, L., A Wide-baseline Matching Library for Zeno, Internship report, 2007.

Garcia, V., Debreuve, E., K-nearest neighbor search: Fast GPU-based implementations and application to high-dimensional feature matching. In: 17th IEEE International Conference on Image Processing (ICIP), 2010, 3757–3760.

Garcia, V., Debreuve, E., and Barlaud, M., Fast k nearest neighbor search using GPU. In: CVPR Workshop on Computer Vision on GPU, Anchorage (AK), USA, 2008.

Goedemé, T., Nuttin, M., &Tuytelaars, T., et al. Omnidirectional vision based topological navigation. In: International Conference on Computer Vision, 74 (3), 2007, 219–236.

Guerrero, J.J., Murillo, A.C., Sagues, C., 2008. Localization and matching using trifocal tensor with bearing-only data. IEEE Trans. Robotics 24 (2), 494–501.

Harris, C. and Stephens, M., A combined corner and edge detector. In: Proceedings of the Fourth Alvey Vision Conference, vol. 15, 1988, 147–151.

Hedborg, J., Skoglund, J., and Felsberg, M.. KLT tracking implementation on the GPU. In: Proceedings SSBA 2007, Linkoping, Sweden, 2007.

Kadir, T., Zisserman, A., and Brady, M., An affine invariant salient region detector. In: Proceedings of the Eighth European Conference on Computer Vision, 2004, 228–241.

Kortenkamp, D., Bonasso, R.P., & Murphy, R.. AI-based Mobile Robots: Case Studies of Successful Robot Systems, The MIT Press, USA, pp. 91–122.

Koseck, Jana, Li, Fayin, Yang, Xialong, 2005. Global localization and relative positioning based on scale-invariant keypoints. Robotics Autonomous Syst. 52 (1), 27–38.

Lowe, D., 2004. Distinctive image features from scale-invariant key points. Int. J. Comput. Vision 60, 91–110.
Matas, J., Chum, O., Urban, M., Pajdla, T., 2004. Robust wide-baseline stereo from maximally stable extremal regions. Image Vision Comput. 22, 761–767.
Mikolajczyk, K., Schmid, C., 2004. Scale and affine invariant interest point detectors. Int. J. Comput. Vision 60, 63–86.
Morel, J.M., Yu, G, 2009. ASIFT: a new framework for fully affine invariant image comparison. SIAM J. Imaging Sci. 2 (2).
Murillo, A.C., Sagues, C., Guerrero, J.J., 2007. From omnidirectional images to hierarchical localization. Robotics Autonomous Syst. 55 (5), 372–382.
NVIDIA. ⟨http://www.nvidia.com/cuda⟩.
Navid, N.V.Cedric, P. . Scene change detection for topological localization and mapping. In: IEEE/RSJ International Conference on Intelligent Robotics and Systems (IROS), 2010.
Nistér, D., 2004. An efficient solution to the five-point relative pose problem. IEEE Trans. Pattern Anal. Mach. Intell. (PAMI) 26 (June 6), 756–770.
A.C. Schultz, W. Adams. Continuous localization using evidence grids. Proceedings of the IEEE International Conference Robotics and Automation, 1998, 2833–2839.
S.N. Sinha, J.-M. Frahm, M. Pollefeys. GPU based video feature tracking and matching. In: EDGE 2006, workshop on Edge Computing Using New Commodity Architectures, Chapel Hill, May 2006.
Sinha, S.N., Frahm, J.-M., Pollefeys, M., Genc, Y., 2007. Feature tracking and matching in video using programmable graphics hardware. Mach. Vision Appl..
Sinha, Sudipta N, GPU_KLT: A GPU-based Implementation of the Kanade-Lucas-Tomasi Feature Tracker, ⟨http://cs.unc.edu/∼ssinha/Research/GPU_KLT/⟩.
Tapus, A. and Siegwart, R.. Incremental robot mapping with fingerprints of places. In: IEEE/RSJ International Conference on Intelligent Robotics and Systems (IROS), pages 2429–2434, 2005.
Tuytelaars, T. and Van Gool, L., Wide baseline stereo matching based on local, affinely invariant regions. In: Proceedings of the British Machine Vision Conference, 2000, 412–425.
Tuytelaars, T., Van Gool, L., 2004. Matching widely separated views based on affine invariant regions,. Int. J. Comput. Vis. 59, 61–85.
Zivkovic, Zoran, Booij, Olaf, Kr̈ose, Ben, 2007. From images to rooms. Robot. Autom. Syst. 55 (May 411–418).
Changchang, W., SiftGPU: A GPU Implementation of Scale Invariant Feature Transform (SIFT), ⟨http://www.cs.unc.edu/∼ccwu/siftgpu/⟩.