# Self-Localization of a Heterogeneous Multi-Robot Team in Constrained 3D Space

Yi Feng
Dept. of Computer Science
The Graduate Center, CUNY
yfeng@gc.cuny.edu

Zhigang Zhu
Dept. of Computer Science
City College of New York
zhu@cs.ccny.cuny.edu

Jizhong Xiao
Dept. of Electrical Engineering
City College of New York
jxiao@ccny.cuny.edu

***Abstract*— This paper presents a new approach to the self-localization among a team of robots working in constrained 3D space of an urban environment. As the base formation, a team of three ground robots and one climbing robot are deployed on ground and on a wall or ceiling, respectively. The three ground robots localize themselves using an existing omnidirectional vision-based method. However, no method exists to uniquely determine the pose of the climbing robot based on the positions of three ground robots in its image and in the world; up to four valid solutions exist using known methods, although only one is genuine. Through careful examination of these methods, two new algorithms for locating the climbing robot are proposed. The first algorithm makes use of the straight line constraint of robot motion and can uniquely determine the pose of climbing robot by moving the climbing robot straight for two small steps. The second algorithm is based on the principle of Bayesian localization method and takes advantage of the motion sensor readings to loose the straight line constraint. This algorithm could continuously determine the climbing robot's pose after the initial pose is obtained. Extensive simulations are conducted to validate the soundness and robustness of the two algorithms. Preliminary real experiments are also carried out to examine the feasibility in applying these algorithms to real robot applications.**

## I. Introduction

### A. Motivation

The advancement in Unmanned Aerial Vehicles (UAVs)[3] and wall-climbing robots [13][19][4][1] has extended the workspace of robots from 2D to 3D. Unlike the UAVs which have full freedom in 3D space; the wall-climbing robots essentially operate in constrained 3D space, i.e., its action space is confined within planar surfaces while the sensing space is 3D, facilitated by the freedom of motion on ground, walls, and ceilings. Employing climbing robots in a robot team in urban operations (both indoors and outdoors) provides both opportunities and challenges. The most profound benefit is that the climbing robots can take vantage positions on a ceiling or wall to gain better views of a scene, ( e.g., a bird's eye view) and to avoid occlusions and obstacles. The challenges lie in the fact that the linear and all simple systems in 2D may become non-linear and much more complicated in 3D, which dramatically increase the complexity of motion planning, localization and navigation problems. Most of the existing methods for multi-robot systems in 2D are no longer valid for the application scenario involving wall-climbing robots.

The above motivation drives us to conduct a series of research on a new framework to deal with this hard yet previously unexplored research domain of multi-robot cooperation in a constrained 3D space.

### B. Self-localization Problem

The application scenario in this paper is stated as follows: a team of heterogenous robots are deployed in a 3D space. Among them a group of four robots, including three wheel-driven ground robots and one climbing robot, generate a basic formation via resource allocation. The three ground robots mutually view each other with panoramic cameras or wide angle cameras. The climbing robot can detect the three ground robots which cannot see the climbing robot conversely due to the limit of camera fields of view in the vertical direction. The objective of our self-localization is to obtain the pose information with six degrees of freedom (DOF) among all the four robots. In [16], Spletzer, *et. al.*provided an algorithm localizing the three ground robots with mutual visibility. In [11][9][6] and [12], the authors attempted to solve the problem of estimating the pose of an overhead camera by viewing three reference points with different approaches. However, the pose of the overhead camera cannot be uniquely determined due to the lack of constraints in such a non-linear problem - there could exist up to four valid solutions (only one of them is genuine). In [14], Quan and Lan's a method with four known points could accomplish the task but with relatively low accuracy, while with five points we could fulfill the task with high accuracy. Obviously, a method requiring fewer known 3D points provided by robots themselves will be more attractive since it reduces the system cost and avoids occlusion problem and sensor noise sources. Since [16] has provided a practical solution for determining 3D poses of the three ground robots, the focus of our self-localization is to solve the pose estimation problem of overhead camera (installed on climbing robot) using only three reference points (ground robots as known landmarks). To the best of our knowledge, there are no existing algorithms that can determine the unique solution using only three reference points.

### C. Related Work

Robot localization problem has been studied extensively in the past decades. Representative work includes the following approaches: deterministic algorithms based on landmark and position tracking [18]; probability based algorithms in continuous domain (Extended Kalman Filter and Markov

Localization) [2][10], and probabilistic algorithms in discrete domain (Monte Carlo Localization)[7][17].
In addition, Spletzer *et. al.*proposed the algorithm of localizing three ground robots in flat space using three panoramic cameras. The three-point camera pose estimation algorithm is also thoroughly studied in a deterministic approach in [11][9][6] and [12], up to four valid solutions. All these works provide the background and the basic elements of algorithms proposed in this paper.

### D. Our Contribution

We propose two algorithms that solve the climbing-robot camera pose estimation problem with three reference ground robots only. The first one is a deterministic algorithm, which uses the constraint of straight line motion of climbing robot by commanding the robot move straight and taking snapshots of the three ground robots three times (a small step each time), thus the three poses of the climbing robot can be uniquely determined. The second one is a probabilistic method inspired by Monte Carlo localization [7][17] algorithms. The method takes advantage of a motion sensor on the wall-climbing robot to roughly record the climbing robot's motion and eliminate the pseudo solutions (among the up-to four valid solutions) by *belief* update of the robot motion. After the initial convergence stage, the genuine solution is identified and then tracked throughout the robot movement. Simulations and real experiments are carried out to verify the soundness and robustness of our methods.
The contribution of this paper lies on four folds. First, the three point camera pose estimation problem, which was proved unsolvable in analytical approach, is solved in both deterministic and probabilistic manners for the first time. Second, the location of the climbing robot could be continuously determined after the initial convergence stage in our probabilistic algorithm, which excels the previous algorithm in [5] that requires multiple movements of ground robots. Third, the simulation extends the accuracy analysis in [11] and a lot of new observations are confirmed in the three point camera pose estimation problem. Fourth, the probabilistic algorithm is computing efficient and can be implemented in real time.
This paper is organized as follows. In Section II, the problem to be solved is formally defined. In Section III, preliminary algorithms in [16] and [6] are briefly reviewed. In Section IV, our deterministic self-localization algorithm is formally proposed. In Section V, the probabilistic self-localization algorithm is described. In Section VI, extensive simulation results and real image based experiments are presented. Section VII arrives at the conclusion of the paper.

## II. Problem formation

Our system is composed of four robots, each mounted with a camera, as illustrated in Fig. 1. The three robots mounted with cameras $C_1, C_2$ and $C_3$ are on the ground, which does not have to be a planar surface. The robot equipped with a camera $C$ is a wall-climbing robot which can move on the ground, climb on walls or stay on the ceiling.

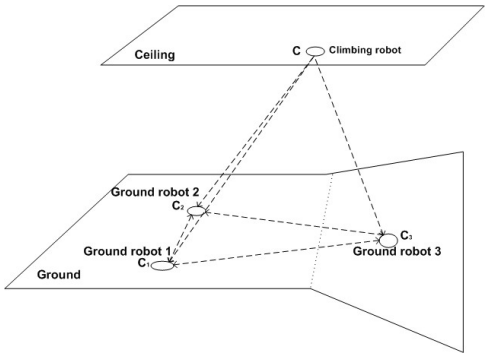



Fig. 1: Four-Robot formation in 3D circumstance

Let $C_1, C_2$ and $C_3$ be panoramic cameras with paraboloid reflecting mirrors and camera $C$ be a perspective camera; let $n \times n$ be the resolution of camera $C$, and $n_i \times n_i$ be equivalent sensor resolution of the panoramic camera $C_i$, $i = 1, 2, 3$. Let $k_i$ be the conic constant of each paraboloid camera mirror. Let $f$ and $f_i$ be the effective focal lengths of camera $C$ and $C_i$, respectively. Let $\alpha$ be the aspect ratio of camera $C$ and let $\alpha_i = 1$ be the aspect ratio of camera $C_i$, $i = 1, 2, 3$. Let $T_i = (x_i, y_i, z_i)^T$ and $T = (x, y, z)^T$ be the position of camera $C_i$ and $C$ in the same world coordinate system, respectively. Let $R_i$ and $R$ be the rotation matrices of cameras $C_i$ and $C$ in the world coordinate system. Let ${}^jT_i$ and $T_i$ be the translations between camera $C_i$ and $C_j$, and $C$ and $C_i$, respectively. Let ${}^jR_i$ and $R_i$ be the rotation matrices between camera $C_i$ and $C_j$, and $C$ and $C_i$, respectively. For any point $X_0 = (x_0, y_0, z_0)^T$, it's position in the image camera $C_i$ and $C$ can be uniquely determined by the parameters above.

**Problem 1:** Let $(u_{ij}, v_{ij})$ be the position of camera $C_j$ in camera $C_i$'s image coordinate system, for all $i, j = 1, 2, 3$ while $i \neq j$. Let $(u_j, v_j)$ be the position of camera $C_j$ in camera $C$'s coordinate. Without loss of generality, assume the distance between camera $C_1$ and $C_2$ is known, which is noted as $\|{}^2T_1\|$. The results to be computed are the rotation matrix ${}^jR_i$ and $R_i$, and translation vector ${}^jT_i$ and $T_i$ for all $i, j = 1, 2, 3$ while $i \neq j$.

In summary, problem 1 is that a wall-climbing robot localizes itself by seeing three ground robots, whose poses are determined by each of them seeing the other two. In the end, the 3D positions and orientations of all the four robots are determined without using any other landmarks in the environment.

## III. Preliminaries

In this section, we will briefly review the two existing algorithms which serve as the basic components of our new algorithms.

### A. Three Robots Localization Algorithm

This algorithm computes the translations and rotations among cameras $C_1$, $C_2$, and $C_3$ with mutual visibility. It is originally completely stated in [16]. Therefore, we will

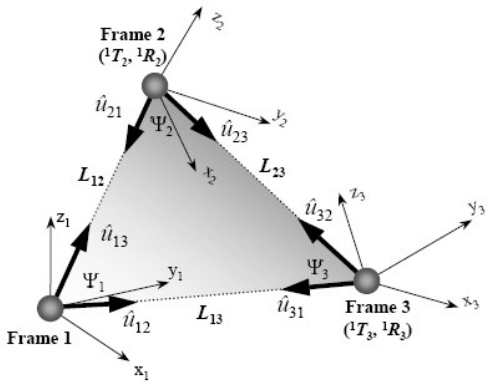



Fig. 2: Three-robot localization (Courtesy [16])

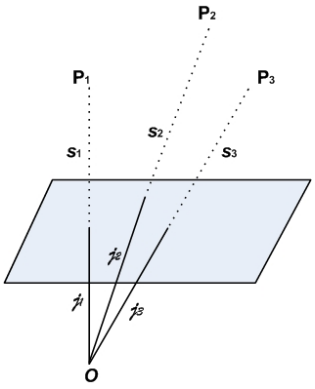



Fig. 3: Three point camera pose estimation. $p_i$ is the point vector from the optical center, $s_i = \|p_i\|$, $j_i$ is $p_i$ scaled by scene depth $z_i$ and focal length $f$.

only present the input and output of the algorithm for notion purpose of this paper. As shown in 2, the three omnidirectional cameras $C_1$, $C_2$, and $C_3$ are pre-calibrated. Therefore, their intrinsic parameters are known. Camera $C_i$'s position in camera $C_j$'s image can be extracted by a simple image processing algorithm given that each robot is color-coded or has active signal (e.g. IR) to be easily detected, for $i, j = 1, 2, 3$ where $i \neq j$. By running the algorithm in [16], the translations ${}^iT_j$ (up to a scale) and rotations ${}^iR_j$ between any pair of cameras $C_i$ and $C_j$ can be calculated. The algorithm can be implemented in real time.

### *B. Three Point Camera Pose Estimation Algorithm*

Firstly, we define the three point camera pose estimation problem.

**Problem 2:** Shown as in Figure 3, let $P_1$, $P_2$, and $P_3$ be three points with known positions in the world coordinates; let $(u_i, v_i)$ be their positions in the image coordinate system of the camera $C$ $(i = 1, 2, 3)$, the problem of 3 Point Camera Pose Estimation is to calculate the pose (position and orientation) of the camera in the world coordinate system.

There are several independent but unfortunately incomplete solutions to this problem. Because of the nonlinear nature of this problem, all independent algorithms offer up to four (at least one) valid solutions among which only one is the genuine solution. The first algorithm [9] was found by Grunert back in 1841. Finsterwalder [6] and Linnainmaa *et. al.*[12] solved the problem independently in 1937 and 1988, respectively. A thorough history of the problem is discussed by Haralick *et. al.*in [11]. After having implemented and examined all these algorithms, we have found that the algorithm by Finsterwalder [6] provides the best accuracy and reasonable running time under our system configuration. Before getting into our algorithms proposed in this paper, this algorithm is narrated in a mathematical manner for completeness and easy reference in the following sections.

As show in Figure 3, the three world points, $p_1$, $p_2$ and $p_3$, corresponding to cameras $C_1$, $C_2$ and $C_3$ on three ground robots are viewed at optical center $O$ of camera $C$, and $p_i = (x_i, y_i, z_i)^T$, $i = 1, 2, 3$. Let $a = \|p_2 - p_3\|$, $b = \|p_1 - p_3\|$, $c = \|p_2 - p_1\|$ denote the side lengths of the three-camera-triangle. Let the focal length of camera $C$ be $f$. The image coordinates $q_i = (u_i, v_i)$, is known from image processing. Meanwhile, $u_i = f\frac{x_i}{z_i}$ and $v_i = f\frac{y_i}{z_i}$, where $f$ is known. The vectors $j_1$,$j_2$,and $j_3$ are the in-camera vectors point from optical center to the points $p_1, p_2, p_3$ and can be represented by $\frac{1}{\sqrt{u_i^2+v_i^2+f^2}}(u_i, v_i, f)^T$, $i = 1, 2, 3$. Let the angles opposite to side $a$, $b$ and $c$ be $\alpha$, $\beta$ and $\gamma$. They could be calculated by $\cos\alpha = j_2 \cdot j_3$, $\cos\beta = j_1 \cdot j_3$, $\cos\gamma = j_1 \cdot j_2$. Let the unknown distances from $O$ to $P_i$ be $s_i$ where $s_i = \|p_i\|$. Since $p_i = s_i j_i$ for $i = 1, 2, 3$, if scalars $s_1$, $s_2$, and $s_3$ are determined, $p_1$, $p_2$ and $p_3$ consequently are determined. The pose of camera $C$ can be estimated accordingly.

By the law of cosines, we have

$$s_2^2 + s_3^2 - 2s_2 s_3 \cos\alpha = a^2 \tag{1}$$

$$s_1^2 + s_3^2 - 2s_1 s_3 \cos\beta = b^2 \tag{2}$$

$$s_1^2 + s_2^2 - 2s_1 s_2 \cos\gamma = c^2 \tag{3}$$

$$\text{Let } s_2 = us_1 \text{ and } s_3 = vs_1 \tag{4}$$

Then it stands

$$\begin{aligned} s_1^2 &= \frac{a^2}{u^2+v^2-2uv\cos\alpha} \\ &= \frac{b^2}{1+v^2-2v\cos\beta} \\ &= \frac{c^2}{1+u^2-2u\cos\gamma} \end{aligned} \tag{5}$$

from which it holds

$$u^2 + \frac{b^2-a^2}{b^2}v^2 - 2uv\cos\alpha + \frac{2a^2}{b^2}v\cos\beta - \frac{a^2}{b^2} = 0 \tag{6}$$

$$u^2 - \frac{c^2}{b^2}v^2 + 2v\frac{c^2}{b^2}\cos\beta - 2u\cos\gamma + \frac{b^2-c^2}{b^2} = 0 \tag{7}$$

By computing (7)$\times\lambda$+(6) we have

$$Au^2 + 2Buv + Cv^2 + 2Du + 2Ev + F = 0 \tag{8}$$

where the coefficients $A = 1+\lambda$, $B = -\cos\alpha$, $C = \frac{b^2-a^2}{b^2} - \lambda\frac{c^2}{b^2}$, $D + -\lambda\cos\gamma$, $E = (\frac{a^2}{b^2} + \lambda\frac{c^2}{b^2})\cos\beta$ are expressed in

terms of parameter $\lambda$.

Take (8) as a quadratic equation of $v$, we can solve it to obtain

$$v = \frac{-(Bu+E) \pm \sqrt{(B^2-AC)u^2 + 2(BE-CD)u + E^2 - CF}}{C} \quad (9)$$

Now we need to solve the parameter $\lambda$. We observe that $\lambda$ should make $(B^2 - AC)u^2 + 2(BE - CD)u + E^2 - CF$ a perfect square of $u$ in order to provide real solutions of $u$ (has physical meanings). If there exists such $\lambda$, (9) could be substitute back to (6) or (7) to solve $u$ and consequently solve $v$.

The value of $\lambda$ produces $(B^2 - AC)u^2 + 2(BE - CD)u + E^2 - CF$ a perfect square should satisfy

$$G\lambda^3 + H\lambda^2 + I\lambda + J = 0 \quad (10)$$

where $G = c^2(c^2\sin^2\beta - b^2\sin^2\gamma)$, $H = b^2(b^2 - a^2)\sin^2\gamma + c^2(c^2 + 2a^2)\sin^2\beta + 2b^2c^2(-1 + \cos\alpha\cos\beta\cos\gamma)$, $I = b^2(b^2 - c^2)\sin^2\alpha + a^2(a^+2c^2)\sin^2\beta + 2a^2b^2(-1 + \cos\alpha\cos\beta\cos\gamma)$, $J = a^2(a^2\sin^2\beta - b^2\sin^2\alpha)$.

Solving (10) provides up to three real roots of $\lambda = \lambda_0$, which determines $A$, $B$, $C$, $D$, $E$ and $F$. We substitute (9) into (7), and then we solve a quadratic equation of $u$. Afterward we take $u$ back to (9) to obtain $v$.

Given $u$ and $v$ solved, we take them back to $s_1$, $s_2$, and $s_3$, the pose of camera $C$ can be estimated. However, we notice that (10) offers up to three real solutions of $\lambda$, in addition, for each solution $\lambda = \lambda_0$, we need to solve a quadratic solution of $u$, therefore, there could be up to six valid solutions. However, it is observed that if there are two or three real solutions in (10), two of them (correspond to the pair of solutions on the quadratic factor component) will provide the same pair of solutions when we solve the quadratic equations of $u$. Therefore, the algorithm will offer up to four valid solutions.

The techniques in this algorithm only concern solving third order and quadratic equation, which has deterministic solution. The computation in this algorithm only evolves deterministic regular operation, thus it can be implemented in a real time manner.

## IV. Deterministic algorithm

Up to this point, we could locate the three ground robot (up to a scale) and offer up to four valid solutions (one of them is genuine) on estimation of the pose of camera $C$. In this section, we will offer a deterministic "linear movement" algorithm that eliminates those pseudo solutions.

The basic idea of this algorithm is very straight forward: if the robot moves along a straight line, the genuine solutions can be picked up by using the linear movement constraint. The algorithm is also very practical since only three snapshots of the ground robot team are needed when the wall-climbing robot is moving on a straight line. The linear motion algorithm is composed of the following steps.

1. The three ground robots localized themselves and then remain stationary. Camera $C$ on climbing robot takes a snapshot on the three ground robots and estimates its own pose, provides up to four valid solutions.
2. Camera $C$ moves once for a short distance, then it retake a snapshot of three ground robots. Camera $C$ redoes three point pose estimation, and obtains another group (up to four) of valid solutions.
3. Camera $C$ moves the second time along the same direction as in 2., takes the third snapshot and obtains the third group of valid solutions (up to four exist).
4. Now we have three groups of valid solutions, which provides up to $4 \times 4 \times 4 = 64$ solution combinations. By using the linear motion constraint of the robot movement, we do steps 5. on all these solution combinations.
5. For each solution combination, determine a line by the first camera pose solution and the third camera pose solution. Calculate the distance of the second camera pose to the line, record it as the *error* of current solution combination.
6. Take the solution option with smallest *error* as the genuine solution.

This algorithm is very simple and straight forward - if the robot moves along a straight line, the genuine solutions should also move along a straight line. On the other hand, pseudo solutions can hardly keep along a straight line. Even though it is hard to provide a mathematical proof of this observation, we have run extensive simulations to validate this conclusion. Details of experiments are discussed in section VI.

## V. Probabilistic Algorithm

It seems that if we could loose the linear motion constraint to smooth motion constraint, the algorithm could be more effective and practical. However, we have observed from our experiments that if the robot moves continuously and smoothly, both the genuine solutions and pseudo solutions follow smooth and continuous paths if real solutions exist. Going back to the three point algorithm in Section III, we could find that the algorithm was designed to solve a fourth order equations system, thus given one of the solutions varies continuously, the other three will also vary continuously given the existence of the real solution. In addition, other three camera pose estimation algorithms in [9][12][11] are all based on similar equations. The properties of all these algorithms prevent us from using the smoothness constraints to eliminate pseudo solutions. Inspired by Bayesian based algorithms (in particular the Monte Carlo localization method), we propose a new probabilistic algorithm to localize the climbing robot.

### A. Overview of the Algorithm

This probabilistic algorithm takes advantage of the motion sensor on the climbing robot to measure and record the action of the climbing robot (i.e., camera C). Meanwhile, the climbing robot estimates its relative position to the ground-robot team by applying the camera pose estimation algorithm via the observation of its vision sensor. The robot verifies its observation with its action using particle filter to decide its location. Unlike the prevous Monte Carlo localization methods which make large scale sampling, our self-localization algorithm only places 4 samples, i.e., four

valid solutions provided by the three-point algorithm. In each iteration, our algorithm updates its *belief* at each location by its observation (vision sensor data) and its action (motion sensor data).

Mathematically, we define the algorithm as follows.

Without loss of generality, let $x_t^{(i)}$, $i = 1, 2, 3, 4$ be the possible positions of climbing robot at time $t$, where superscript $(i)$ denotes the four valid solutions by camera pose estimation algorithm. Let $belief(x_t^{(i)})$ be the belief value of the climbing robot is at position $x^{(i)}$ at time $t$. $belief(x_t^{(i)})$ is normalized at any time $t$ by normalizer

$$\pi(t) = \sum_{i=1}^{4} belief(x_t^{(i)}) \tag{11}$$

Let $o_t$ be the observation of the climbing robot at time $t$. In our specific system setting, $o_t$ is defined as the estimated pose change of climbing robot at time $t$. We have observed in our experiments that if the climbing robot moves continuously, all the valid solutions of estimated climb robot poses also vary continuously. Meanwhile, the valid solutions are sufficiently far apart from each other, therefore, we could trace each valid solution by the nearest neighbor in the previous observation $o_{t-1}$ give its existence. Otherwise we will ignore the solution in that it is obviously an invalid pseudo solution. For simplicity, we defined $om_t$ as the observed motion from time $t-1$ to time $t$ by vision system and nearest neighbor trace. If we could not find a reasonable close nearest neighbor of $o_t$ from $o_{t-1}$, we will record $om_t$ as $\infty$.

Let $a_t$ be the action (motion sensor data) of the climbing robot between time $t-1$ and $t$. In our problem formation, it is a 3D vector, which is of the same format as $om_t$.

At the very initial stage $t = 0$, $belief(x_0^{(i)} = \frac{1}{4}$ given four valid solutions upon existence.

At each time $t$, we update the $belief(x_t^{(i)})$, $(i = 1, 2, 3, 4)$, by

$$belief(x_t^{(i)}) = belief(x_{t-1}^{(i)}) \times f(om_{t-1}^{(i)}, a_{t-1}^{(i)}) \tag{12}$$

in which $f(om, a)$ is the *likelyhood function* calculating the similarity *observed motion* and *action*. After each update, the $belief(x_t^{(i)})$ should be normalized across all possibilities $(i)$. When the *belief* of the climbing robot at one position reach a threshold of confidence, we will take this position as the genuine robot track and keep tracking. During this stage, we still update the *belief*on all four positions. Once the belief on this position reduces to the threshold, we will relocate the climbing robot position.

### B. Likelyhood Function

The likelyhood function plays an important role in the implementation of our algorithm. It should well categorize the match and unmatch of *action* and *observed motion*. We developed two options for the likelyhood function.

**Definition 1:** The simple likelyhood function is the normalized inner product of two 3D vectors, $a$ and $om$.

This simple function will be great given identical $a$ and $om$, while noticeably small if one or more dimensions on the two vectors remarkably differ from each other.

**Definition 2:** A complete likelyhood function on two 3D vectors $a$ and $om$ is defined as,

$$f(a, om) = \|a - om\| + \lambda \arccos \frac{a \cdot om}{\|a\|\|om\|} \tag{13}$$

In this likelyhood function, the difference between $a$ and $om$ are calculated by both the translation difference and heading difference. The parameter $\lambda$ plays a role in balancing the translation and heading differences. In the three point pose estimation algorithms, we notice that the shift of one solution could result in shift of other solutions in different rate due to the high order of the equations, therefore, we consider a relatively large $\lambda$ in this function to address the importance of motion consistency in moving direction. This is further verified in our experiment in section VI.

### C. Monte Carlo Algorithm

Now we give the formalized Monte Carlo algorithm in pseudo code.

```
MCA(A, OM, BELIEF_THRESH)
/*  A: the vector of actions;OM : the vector of observed motion;
BELIEF_THRESH: threshold of determine the robot position*/
1.  double belief[4]={0.25 0.25 0.25 0.25}
2.  for t = 1 to n
3.      for each valid solution track i
4.          belief[i]=belief[i]*f(a[t],om[t]);
5.      end
6.      normalize(belief[]);
7.      if(max(believe)>BELIEF_THRESH)
8.          current_location=x[max];
9.  end
```

It can be noticed that the complexity of our probabilistic self-localization algorithm is low - each iteration only takes small amount of time to compute. Quite different from large scale Monte Carlo localization method, 4 sample points are sufficient to solve the self-localization problem, which makes it feasible in real-time applications.

## VI. Experiments

We have conducted extensive simulations on both algorithms and carried out experiments on real image data.

### A. Deterministic Algorithm Simulation

The simulation is conducted in Matlab. In the simulated environment, we modeled three ground cameras with omnidirectional views, all with a resolution of $512 \times 512$. We modeled the climbing robot camera as a perspective camera with a resolution of $640 \times 480$. By using catadioptric geometry (of the omnidirectional cameras) and perspective geometry (of the perspective camera), given the ground truth data of all the camera poses (positions and orientations), we obtained the ground truth data of the camera $C$'s or $C_i$'s position in the view of the camera $C_j$. After acquiring such data, we add a Gaussian noise of $N(0, 0.5)$ pixels to the data at a random direction (uniformly distributed across 360 degree of the sensor) in order to test the algorithm robustness.

*1) Experiment Setup:* We setup our robot work space as a $3 \times 3 \times 3$ meters cube. We ran the following experiment for 1000 times.

1. Randomly generate the positions of three "ground" cameras $C_1$, $C_2$ and $C_3$ in sub-cubes with diagonals $(0, 0, 0) - (1, 1, 1)$, $(0, 2, 0) - (1, 3, 1)$ and $(2, 1, 0) - (3, 2, 1)$, respectfully, with uniform distributions.
2. Randomly generate orientation, i.e., the pitch, tilt and

| **Camera** | Relative error | Standard Dev. |
|---|---|---|
| $C_1$ (Ref) | 0 | 0 |
| $C_2$ | 1.7e-4 | 6.1e-5 |
| $C_3$ | 1.6e-4 | 5.1e-5 |
| $C$ | 5.7e-4 | 1.3e-4 |

TABLE I: Simulation Results: the relative mean error of positions using straightness movement algorithm

| **Algorithm** | Running Time |
|---|---|
| Deterministic | 1.7 ms |
| MC (each update) | 0.35 ms |

TABLE II: Simulation Results: time complexity

yaw angles of the cameras $C_1$, $C_2$, $C_3$ and $C$ to be evenly distributed within $[0, \pi/6]$. We make the camera $C$ as a wide angle camera with FOV of $2/3\pi$ to guarantee the cameras $C_1$, $C_2$ and $C_3$ appear within its FOV.

3. Randomly generate the position of camera $C$ (on the "wall-climbing" robot) in the sub-cube diagonal with $(1, 1, 2)$ and $(2, 2, 3)$, which is a cube right under the center of the ceiling, to simulate ceilings of different heights.

4. Move the camera $C$ alone the direct of $(\Theta, \Phi)$ for distance $X$ twice. The random variables $\Theta$ stands the tilt, $\Phi$ stands for pitch. The random variables follow distributions as $\Theta, \Phi \sim U(0, \pi/6)$, $X \sim N(0.4, 0.2)$. The positions are added with a Gaussian noise of $N(0, 0.001)$ meter, perpendicular to the direction of $X$.

5. On the above formation and movement, run the linear movement algorithm, setting the *error* threshold to be $0.003$ meter.

*2) Simulation Results:* After running the simulation, we found some general statistics of the simulation. First, during the 1000 simulation, 997 of them offers correct results, which means within an relative error of $1e-3$ meters out of a scale of three meters, the algorithm could identify the correct positions of the climbing robot in almost all the cases. We further look into the three problematic instances, and find the failure are caused by the singularity of three point pose estimation problem. This illustrates that our algorithm is sound in all valid experiments.

Table I indicates the statistics of the 997 plausible solutions. It shows that by employing the co-linearity verification complemented by moving climbing robot along a line to obtain three samples, our method can generate unique pose estimation of climbing robot with high accuracy at an relative error level of $1e-4$.

Figure 4 is the illustration of how our deterministic algorithm can robustly pick up the genuine solution. The example is randomly selected from our 997 successful simulations. The three red asterisks represent the position of three ground robots. The pattern of a circle with a line stands for the camera pose - located at the circle and pointing from the circle along the line. The three thick-line patterns, with blue, green and red, respectively, are the genuine (also the estimated) positions of climbing robot. The thin-line patterns stand for those pseudo solutions. All the (genuine and pseudo)

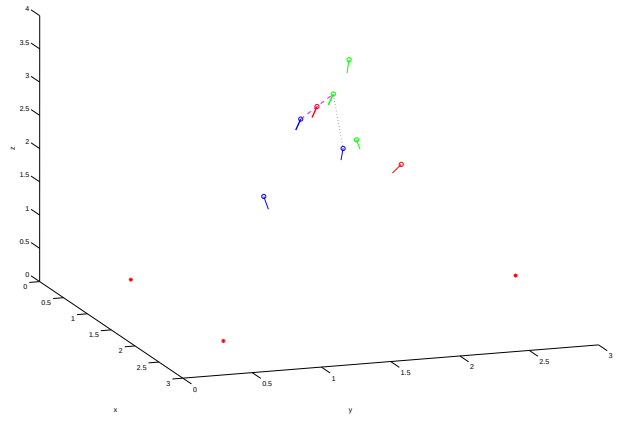


Fig. 4: Illustration of simple algorithm

solutions of the three camera positions are categorized by their colors. Any combination of red, green and blue will form a valid solution combination The magenta dash line illustrated the straightness of the genuine solutions in which the 2nd genuine position lies exactly on the line connecting the 1st and the 3rd ones, and the black dot line stands for the combination of solution combination with second smallest *error*, where the 2nd position's estimation (in red) lies far away from the line connecting the 1st (in blue) and the 3rd (in green) ones. Note that in this specific case, the *error* of the genuine solution is $2.04e-4$ meters and the second smallest *error* (dark dot line from the thick red solution) is $0.379$ meter. In all our $997$ successful runnings, the *average error* of genuine solution is only $3.19e-4$ meters with a *maximum error* of $8.53e-3$ meters. The *average error* of the second best solutions (with second smallest errors) is $0.771$ meter, and the *minimum error* of the second best solution is $0.293$ meters. We could observe that the genuine solutions can be obviously distinguished from pseudo solutions.

### *B. Probabilistic Algorithm Simulation*

*1) Simulation Setup:* We build our environment in Matlab. The work space of ground robots is a $3 \times 3$ meters field. We built two types of ceiling for the climbing robot. Type I is a horizontal ceiling at the height of 3 meters. Type II is a slope ceiling of $45$ degree. It starts from a height $2$ meters at one side and stop at the other side at height of $5$ meters. We let the robot moves at a velocity of $0.2$ meter per second. The algorithm updates at a rate of $10$Hz. We command the robot move at the following three curves starting from the center of the ceiling in the ceiling plane. The original point of the ceiling coordinate is also set at the center of the ceiling.

Curve 1 (straight line): $y = 0.5x$; Curve II (Sine curve): $y = \sin(2x)$; Curve III (Quadratic): $y = x^2$.

The ground robots are arbitrarily set at position $(0.5, 0.5, 0.3)$, $(0.5, 2.5, 0.4)$, $(2.5, 1.5, 0.2)$.

Because we already verified the accuracy of our approach in the simulation of the deterministic linear movement

| Curve | Ceiling I | Ceiling II |
|---|---|---|
| Curve I | 6.93 | 7.31 |
| Curve II | 11.22 | 19.47 |
| Curve III | 8.97 | 14.15 |

TABLE III: Average number of updates to converge on 100 simulations

algorithm in identifying the genuine solutions, the main purpose of the simulation of our probabilistic algorithm is to compare how well the two methods converge to the genuine solutions. We set the threshold of belief at $0.95$. We set the two consequent solutions to be on the same track given their position are closer than $0.05$ meter and the dot product of their normal vector are greater than or equal to $0.85$. We also set the first update interval as $0.3$ second instead of the regular $0.1$ second for the reason that we have stated in the previous section. We run the simulations on both types of ceilings and with the climbing robot moving along all the three types of curves for 100 times. We add a Gaussian noise of a standard deviation $15\%$ on all three dimensions of the motion sensor ($a_t$) to ensure the robustness of our algorithm in that motion sensor are usually of high noise. We also put the noise in image data as we did in previous simulations. The experiment terminates once the robot reaches the boundary of the ceiling. The measurement of algorithm convergence is the average number of updates it takes to reach the belief threshold.

*2) Simulation results:* As indicated in Table III, we could find it take about 1 to 2 seconds for the robot to converge to its genuine solution. A fact to mention is that after the algorithm converges to the genuine solution, it never have a belief under the threshold. This implies the algorithm could keep on the correct track after convergence. We also note that straight moving trajectory converges faster than high order and sinusoid trajectory, this is because linear party of a nonlinear system is easier to detect. In addition, we find that horizontal ceiling case converges faster than the slope ceiling case, which indicates working on two dimensions share the same benefit as linear trajectory.

*3) Time Complexity:* Table II indicates the computational complexity. Note that running time includes the time for generating random sample data and camera model.

### C. Preliminary Real Experiments

To further test our algorithms in real scenarios, we also conducted experiments on real images. Real tests of the algorithms on autonomous multi-robot team as shown in Figure 7 (b) is under construction.

*1) Experiment Setup:* We use one panoramic camera (Remote Reality NetVision 360) and two perspective cameras (Logitech QuickCam Pro Series) as the vision sensors of the three ground robots. We did not mount the cameras onto the robots, partially for obtaining easier "ground-truth" data measurements of the real positions of the cameras for evaluating our algorithms. We use another perspective camera (Logitech QuickCam Pro Series) as the vision sensor

(a) EL panel: on (b) EL panel: off

Fig. 5: EL panels

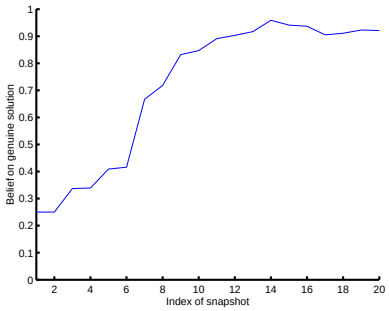


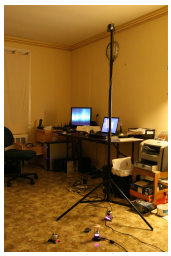

Fig. 6: The belief on genuine solution across the update in real experiment

of the climbing robot. In order to ease the detection of other cameras in each camera's image, we use active light markers on the camera. For localizing the optical center of each camera in images, we wrap a EL Panel around each camera, as shown in Fig. 5. It is a blinking marker and could be bent into any shape as the active vision marker. By controlling blinking of the marker, we could detect other cameras by background substraction. We mounted the overhead camera on a light stand, it moves at the same height and a plumb line was used to measure its exact movement. The experiment setup is shown in Figure 7 (a).

We tested out both our algorithms by taking a series of snapshots. In the deterministic algorithm experiment, we had the overhead camera move along a straight line, at a height of 2 meters, each movement about 20cm, for three snapshots. In our probabilistic algorithm experiment, we controlled the overhead camera to move smoothly on a curve at a height of 2 meters, about 5cm each time, for 20 steps, and record the trajectory on a piece of paper by the plumb line.

The evaluation of the deterministic algorithm is the accuracy of the position of the overhead camera. The evaluation of the Monte Carlo algorithm is to see how the *belief* on a genuine position grows along the twenty snapshots (updates).

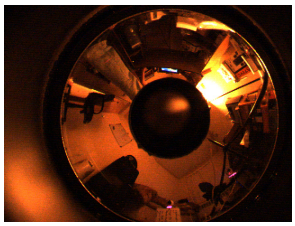
*2) Experiment Results:* In our current experiments, the measurement on ground truth data is of accuracy at 1 cm. The estimated position of overhead camera is of $(2, -4, 3)$cm away from real camera position. This is also the identified genuine solution out of the four valid solutions. Fig. 6 indicates the belief on camera real position as the updates continuous in our Monte Carlo algorithm. It appears to converge well as the overhead camera moves. Fig. 7 illustrated our preliminary experiment setting and the image



(a) Preliminary Experiment Setting

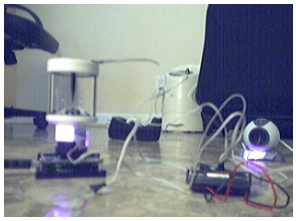
(b) Final Experiment Setting

(c) View from Camera $C$



(d) View from Camera $C_1$

(e) View from Camera $C_2$



(f) View from Camera $C_3$

Fig. 7: Experimental Settings and Views from three different types of cameras. The camera targets are the bright pink EL panels

from different cameras, including how the EL panel appears in these images.

The real experiments successfully examined the feasibility of our algorithms.

## VII. Conclusion and Discussion

In summary, the problem solved in this paper is for a wall-climbing robot to localize itself by seeing three ground robots, whose poses are determined by each of them seeing the other two. Consequently, the 3D positions and orientations of all the four robots are determined without any additional landmarks in the environment. We have presented two algorithms that solve the three point pose estimation problem in an engineering approach. The direct result of this work is a practical achievement of robot self-localization in 3D space. The two algorithms we proposed address different aspects of the 3D self-localization problem. The deterministic linear movement algorithm makes use of the linear motion constraint to solve the multiple solution problem of a nonlinear system. The probabilistic algorithm is based on the Monte Carlo method and looses the linear motion constraint to smooth motion. The algorithm only takes 4 samples to solve the self-localization problem, thus making the real-time implementation possible. Our extensive simulations indicate that the probabilistic algorithm works effectively and robustly in different robot moving trajectories and different surface types. Since we assume that the ground surface and ceiling surface are not necessarily flat, the algorithm proposed in this paper works in any 3D space although the algorithms is developed for the particular case of wall-climbing robots in constrained 3D space. Thus the proposed algorithms can be applied to more general applications, such as UAVs. Our real experiments, though preliminary, examined the feasibility of our algorithms in real application scenarios. Experiment on real robot system is underconstruction.

## References


[1] A. Asbeck, S. Kim, W. Provancher, M. Lanzetta. Scaling Hard Vertical Surfaces with Compliant Microspine Arrays. In *Online Proceedings, Robotics: Science and Systems I*, June 8 11, 2005, MIT, http://www.roboticsproceedings.org/.

[2] W. Burgard, D. Fox, D. Hennig, and T. Schmidt. Estimating the absolute position of a mobile robot using position probability grids. In *Proc. of AAAI*, Menlo Park, August 1996.

[3] Patrick Doherty, Gosta Granlund, Krzystof Kuchcinski, Erik Sandewall, Klas Nordberg, Erik Skarman, and Johan Wiklund. The witas unmanned aerial vehicle project. In *Proceedings of IEEE ROBIO*, pages 246–250, 2005.

[4] M. Elliot, W. Morris, and J. Xiao. City climber: a new generation of wall-climbing robots. In *Video Proceedings of ICRA*, 2006.

[5] Y. Feng, Z. Zhu, and J. Xiao. Heterogeneous multi-robot localization in unknown 3d space. In *Proceedings of IROS*, 2006.

[6] S. Finsterwalder and W. Scheufele. Das Rückwärtseinschneiden im Raum. *Sebastian Finsterwalder zum 75.Geburtstage*, pages 86–100, 1937.

[7] D. Fox, W. Burgard, F. Dellaert, and S. Thrun. Monte carlo localization: Efficient position estimation for mobile robots. In *Proc. of AAAI*, Orlando, FL, 1999.

[8] D. Fox, W. Burgard, and S. Thrun. Markov localization for mobile robots in dynamic environments. *Jounal of Artifical Intelligence Research*, 11:391–427, 1999.

[9] J. A. Grunert. Das pothenotische problem in erweiterter gestalt nebst bber seine anwendungen in der geodäsie. *Grunerts Archiv für Mathematik und Physik*, 1:238–248, 1841.

[10] J. S. Gutmann and C. Schlegel. Amos: Comparison of scan matching appraoches for self-localization in indoorenveronment. In *Proc. of the First Euromicro Workshop on Advanced Mobile Robots*, 1996.

[11] R. M. Haralick, C. N. Lee, K. Ottenburg, and M. Nolle. Analysis and solutions of the three points perspective pose estimation problem. In *Proceedings of CVPR*, pages 592–598, 1991.

[12] S. Linnainmaa, D. Harwood, and L. S. Davis. Pose estimation of a three dimensional object using triangle pairs. *IEEE Transactions on PAMI*, 10:634–647, 1988.

[13] C. Menon, M. Murphy, and M. Sitt. Gecko inspired surface climbing robots. *in Proceedings of the IEEE ROBIO*, 2004.

[14] L. Quan and Z. Lan. Linear n-point camera pose determination. *IEEE Transactions on PAMI*, 21:774–780, 1999.

[15] S. I. Roumeliotis and G. A. Bekey. Bayesian estimation and kalman filtering: A unified framework for mobile robot localization. In *Proceedings of ICRA*, pages 2985–2992, San Francisco, CA, 2000.

[16] J. Spletzer, A. K. Das, R. Fierro, C. J. Taylor, V. Kumar, and J. P. Ostrowski. Cooperative localization control multi-robot manipulation. In *Proceeding of IROS*, volume 2, pages 631–636, October 2001.

[17] S. Thrun, D. Fox, W. Burgard., and F. Dellaert. Robust monte carlo localization for mobile robots. *Artificial Intelligence*, 128:99–141, 2001.

[18] G. Wei$\beta$, C. Wetzler, and E. V. Puttkamer. Keep tracking of position and orientation of moving indoor systems by correlation of range-finder scans. In *Proc. of the International Conference on Intelligent Robots and Systems*, pages 595–601.

[19] J. Xiao, A. Sadegh, M. Elliot, A. Calle, A. Persad, and H. M. Chiu. Design of mobile robots with wall climbing capability. In *Proceedings of the IEEE/ASME ICAIM*, pages 438–443, July 2005.