The 7th Annual IEEE International Conference on
Cyber Technology in Automation, Control and Intelligent Systems
July 31-August 4, 2017, Hawaii, USA

A Global Localization System for Mobile Robot Using LIDAR Sensor

Liqian Feng, Sheng Bi*, Min Dong, Fating Hong, Yuhong Liang, Qinjie Lin and Yunda Liu

South China University of Technology

Abstract—**This paper demonstrates a set of approaches addressing mobile robot localization problems. The approaches can operate efficiently in 2D occupancy grid map created by robot which is equipped with LIDAR sensor. Given a complex environment and a known map, the robot could be placed at anywhere initially and the global localization system described in this paper can be applied to startup stage in which robot can realize its initial location and orientation in the map without any movement. In the following stage, the system can track the moving robot's position in real time. Thus as long as the robot start the navigation process, the robot system knows where the robot is at the beginning and the robot can be navigated to a reachable place directly. Combined with other sensors, such as cliff sensor, the robot can perceive the kidnapped problem, and then relocate the position to recover from global localization failures. The experimental results show that compared with appointing the initial position and orientation manually, it costs seconds in startup stage to figure out the robot's initial pose in the map, and the system can update the robot's position in motion in real time.**

I. Introduction

The robot localization is a fundamental problem in robotics. It is essential for a robot system to get the information of robot's pose that consists of position and orientation. Mobile robot using a variety of sensors can construct the map of surrounding environment. In robotics, there are sorts of maps, such as a graph-like topological map, an occupancy grid map, a manually constructed 2-D metric layout, which are suitable for different use. In localization problem, it is assumed that the map of the environment in robot system is complete and accurate. Meanwhile noise could occur in practice. It could result from the sensors' performance parameter; the motors of the robot could skid on the floor [1]. So, uncertainty is a very important issue, which makes the location unconvincing [2]. A robot equipped a LIDAR sensor can construct a 2D occupancy grid map expediently, and addressing the global localization problem in that map is the focus of this paper.

In Robot Operating System (ROS), the Adaptive Monte Carlo Localization algorithm uses a particle filter to track the position and orientation of a robot against a known 2D map. This approach can track robot's pose in real time. And to cope with global localization problem, after the robot moving a certain distance, the particle would converge in probability, and then the robot would be labeled at the most probably place in the map [3].

Liqian Feng, Sheng Bi, Min Dong Fating Hong, Yuhong Liang, Qinjie Lin and Yunda Liu are with the School of Computer Science&Engineering, South China University of Technology, Guangzhou, 510640. Sheng Bi is also recently with Postdoctoral Fellow in Emerging Technologies Institute, The University of Hong Kong. (corresponding author: Sheng Bi, e-mail: picy@scut.edu.cn)

In practice, using the above approach, before the robot is ordered to move, it has no idea about its pose. So if people enable a robot remotely and cannot order it to move controlled, the automatic motion could result in impact or tumble.

In the novel global localization system mentioned in this paper, the robot can figure out its pose in seconds during the startup stage without motion. Furthermore, people can set goals in the map directly as long as the robot is enabled with the map imported and then the robot system also can track the robot in real time.

In this paper, Section II presents the traditional localization method in ROS as well as its shortages and the interface, with which the programs in the novel system can transmit data. Section III describes the principle of the novel approach, the details, and the whole global localization system combined with multiple algorithms. There are some experiments in various room with different environment. We test the reliability and efficiency compared with the traditional method, and the results are exhibited in Section IV. Finally, Section V discusses the conclusions and the work in future.

II. Procedure for Paper Submission

A. Background

ROS is a fast developing intelligent robotic applications development framework, which support for a large number of sensors [4]. ROS can be installed on Linux and the sensors can receive and dispute message through serial port, which is a relatively reliable channel. In this paper, we use the Kobuki robot with a 2D LIDAR sensor. The Kobuki robot is a two-wheel based mobile robot, which has three bump sensors to detect the impact, several cliff sensors to notice wheels separating from the floor, an imperfect odometer and so on. In addition, as is shown in Fig.1, we install a low-cost LIDAR sensor above the robot. We use a Mini PC installed Linux as the platform. The LIDAR sensor provides the range information of the surrounding environment. When the LIDAR turns a circle, it sends 360 distance data which describes the surrounding from 0 degree to 359 degree respectively.

978-1-5386-0489-2/16/$31.00 © 2017 IEEE

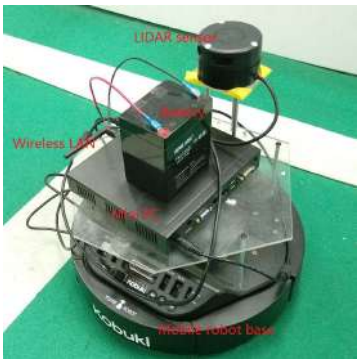


Fig.1. the robot platform and the LIDAR sensor

Furthermore, the ROS can display a graphical user interface in a remote terminal device transmitting data by net. So people can control or only monitor the robot in wireless and remotely. Thus the technique can be used in the house, warehouse management or some other service situation.

B. *Adaptive Monte Carlo Algorithm in 2D Occupancy Grid Map*

Given a mobile robot in a room, the most basic action is moving from a point to another specific point. And there are a large number of algorithms, which based on map to avoid obstacles, such as ceiling visual codes method [5], DWA algorithm [6], and polygonal approximation [7] and so on. People prefer the metric map especially the 2D occupancy grid map in that the information is more detailed and intuitive. This map shows the shape of the environment, shown in Fig.2, which looks like a miniature.

In ROS, the values of 2D occupancy grid map are divided into three categories: occupied, free, and unknown. The occupied (black) pixel means obstacle reflecting the laser, while the free (white) area represent where the laser can pass through. The unknown (gray) space can be seen as the outside the wall. The robot's model can be labeled at the miniature in remote device according to the localization system so that the user would be aware of whether the robot runs normally.

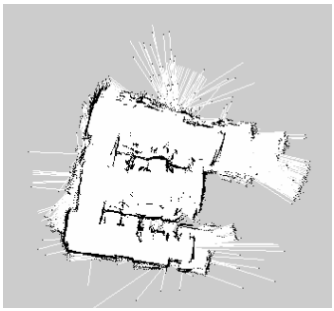


Fig.2. a 2D occupancy grid map of the laboratory

This map is a binary image file, and the pixels are specified in row-major order, which describes the real world using a metric coordinate space. And there is a recorded pixel in the map, which is considered as the origin of the world coordinate. By default, the resolution of the pixel is 5cm. It means every pixel represents a 5cm × 5cm area. As is shown in Fig.3, the LIDAR sensor uses a polar coordinate space to indicate the metric range information.

The Adaptive Monte Carlo Localization package is built in ROS. This program publishes the robot's pose to the node named "amcl_pose" by a certain frequency when the robot is moving [8]. The pose consists of the position coordinate and the orientation in the metric world coordinate to indicate where the robot is, which can be defined as $pose(x, y, \theta)$. In Fig.1, the LIDAR is installed steady above the robot, so the $pose(x, y, \theta)$ describes the origin of the LIDAR's polar coordinate space in the world coordinate, which is a relationship of coordinate space transformations.

Fig.3. the internal polar coordinate space of the LIDAR

Initially, when people start the navigation process with Adaptive Monte Carlo Localization algorithm, the ROS would set the robot's model at the origin in the world coordinate by default though in practice the robot might not be placed at there, which is shown in Fig.4 and Fig.5.

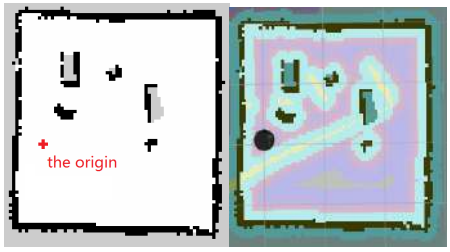


Fig.4. the robot model in the map

Fig.5. the robot in reality

According to Adaptive Monte Carlo algorithm, there are a large number of particles in the map. When the odometer records the motion of robot, the particle converge in probability.

As is shown in Fig.4 and Fig.5, at the beginning, the system did not know the robot's real pose in reality. When we moved the robot by remote controller, which is shown in Fig.6, the particle converged incorrectly. Due to the symmetry, the system believed that the robot was at the upper left in Fig.7. Obviously, the robot was at the lower right in reality.

Fig.6. the route we moved the robot

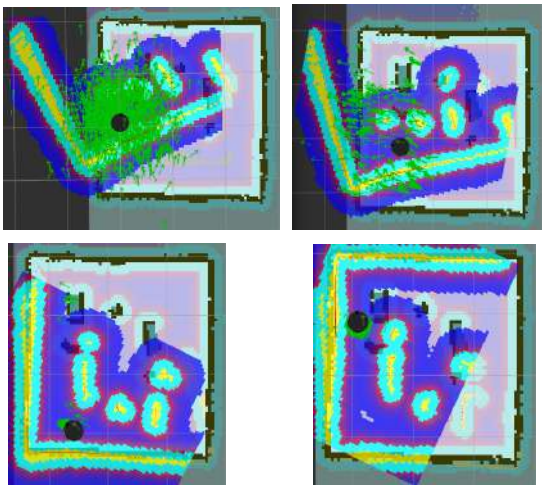


Fig.7. the particle converged in incorrect area

To cope with the issue, the ROS provides a node named "initial_pose". It is an interface in which people can appoint a pose in map manually as the robot's initial pose. In other words, people publishing the initial pose to ROS is to change the default initial $pose(0,0,0)$ into another $pose(x_{real}, y_{real}, \theta_{real})$, which is closed to the real pose in the map.

In Fig.8, the same situation as the Fig.5, we set an initial pose manually. In other words, we can click an upper right pixel in map and appoint an approximate orientation, which is similar to the initial pose in reality. As shown in Fig.9, we appointed the same goal in Fig.6. The pose would be correcting in motion. So in navigation process, robot's pose can be tracked in real time. It is an imperfect method to deal with the global localization problem because it requires human intervention.

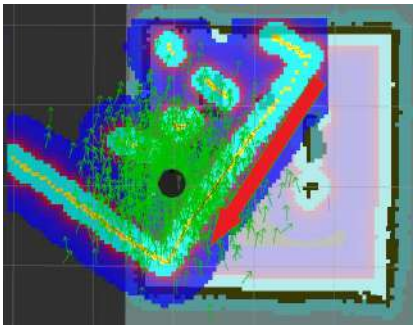


Fig.8. set the initial pose manually

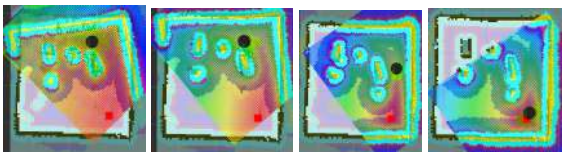


Fig.9. Adaptive Monte Carlo algorithm correct and track the pose

III. THE NOVEL APPROACH

A. Coordinate Space Transformations

The novel global localization system has the key function that robot can figure out its "initial pose" at the beginning according to the 2D LIDAR data and the environment without moving and manual acting.

As is mentioned above, the $pose(x, y, \theta)$ means the origin of the LIDAR's polar coordinate space in the world coordinate and the LIDAR sensor sends 360 distance data describing the surrounding from 0 degree to 359 degree respectively every lap.

We define $dist_i$ as the distance data at i degree counterclockwise around the polar axis, which is mentioned in Fig.3 left. In ROS, the map file describes the *Width, Height* of the map as well as the world coordinate $(X_{lowerleft}, Y_{lowerleft})$ of the lower left pixel in the map. The resolution is 0.05m. The x axis of the world coordinate points to right of the map and the y axis points to up. By default, if the robot is in $pose(0,0,0)$, the polar axis of the LIDAR points to up.

We use a *Height* × *Width* array **m** describes the map. So given a $pose(x, y, \theta)$, every distance datum $dist_i$ from the LIDAR can be projected into the **m** defined as $LIDAR_{x,y,\theta}(height_i, width_i)$, which is shown in (1) and (2).

$$height_i = \frac{-dist_i \cdot \cos(i-\theta) + Y_{lowerleft} - y}{resolution} + Height \tag{1}$$

$$width_i = \frac{dist_i \cdot \sin(i-\theta) - X_{lowerleft} + x}{resolution} \tag{2}$$

So, if we set the robot's initial pose at where the robot is in the real world, every point created by LIDAR sensor would be projected in occupied pixel in ideal case. Otherwise, the majority of points would be in the free or unknown area, which is away from the occupied pixels. The situation is based on an unchanged environment without any dynamic object.

B. Score Model

We come up with a map matching principle. We use the real world environment represented by LIDAR data to match the occupied pixels in map and choose the most accordant position and orientation as the initially pose.

To judge the matching, we can define the $PoseScore(x, y, \theta)$, which describes a pose chosen as the initial pose.

Every $LIDAR_{x,y,\theta}(height_i, width_i)$, for i from 0 to 359, is a pixel in the map. And considering every $LIDAR_{x,y,\theta}(height_i, width_i)$, we define the $Po\,\text{int}\,Score_{x,y,\theta,i}$ as the Manhattan Distance from the pixel to a nearest occupied pixel in the map. For every $pose(x, y, \theta)$, we calculate the summation of $Po\,\text{int}\,Score_{x,y,\theta,i}$ as the $PoseScore(x, y, \theta)$, which is shown in (3).

$$PoseScore(x, y, \theta) = \sum_{i=0}^{359} Po\,\text{int}\,Score_{x,y,\theta,i} \quad (3)$$

For a pose, after coordinate space transformations, if every point of sensor data is located exactly at or near an occupied pixel, which has a low score, it means that the sensor's data matches the map well and this pose chosen in the map would be approximately where the robot is located in real world. So the global localization system should choose $pose(x, y, \theta)$ which get lowest $PoseScore(x, y, \theta)$ as the most possibly initial pose and publish it to ROS.

In basic algorithm, we enumerate every pixel in the map as well as 360 orientation in every degree; for each pixel in enumeration, we assume that the corresponding position combined with the orientation is the initial $pose(x, y, \theta)$ and figure out its $PoseScore(x, y, \theta)$.

Likewise, $PoseInMap(height, width, \theta)$ is defined as the chosen pixel and the orientation. The corresponding $LIDAR_{x,y,\theta}(height_i, width_i)$ is shown in (4) and (5).

$$height_i = \frac{-dist_i \cdot \cos(i + \theta)}{resolution} + height \quad (4)$$

$$width_i = \frac{-dist_i \cdot \cos(i + \theta)}{resolution} + width \quad (5)$$

And $pose(x, y, \theta)$ can be transferred by $PoseInMap(height, width, \theta)$, which is shown in (6) and (7).

$$x = X_{lowerleft} + width \cdot resolution \quad (6)$$

$$y = Y_{lowerleft} + (Height - height) \cdot resolution \quad (7)$$

The *Height* in (7) is height of the map file.

While the accuracy is acceptable, this method is time-consuming.

In traditional grid localization method, the map is divided into grids, and each grid cell stores the probability that the robot is in this cell [9] [10]. The accuracy depends on the size of grid cell [9]. The more accurate, the more expense of computational costs. On the other hand, with a coarse grid, the additional information loss through the discretization negatively affects the filter [1].

This matching method is not the only approach applied in the novel system. We need not a great precision pose in that stage. An approximate result, which is published as the initial pose to ROS would not cause fault in practice because the Adaptive Monte Carlo Localization approach can be applied to the next stage when robot is moving.

To save time and attain an approximate solution, we need not consider every pixel in the map. So every 4 × 4 pixels are combined into a grid. Furthermore, we enumerate the orientation every 15 degree instead of every 1 degree. Thus the run-time is significantly reduced.

C. *Further Optimization*

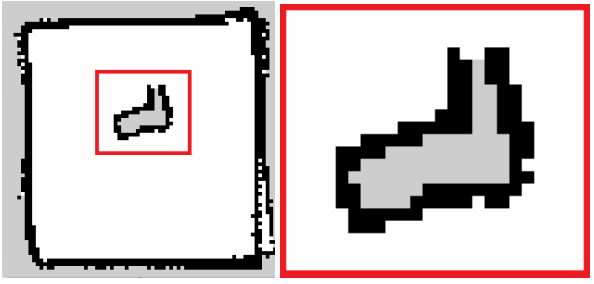
A rapid pre-calculated method can be used to calculate $Po\,\text{int}\,Score_{x,y,\theta,i}$, because it is the Manhattan Distance from the pixel to a nearest occupied pixel in the map, which is a property of a pixel and not relevant to the $pose(x, y, \theta)$.So we can define a *height* × *width* matrix **pointscore** storing $Po\,\text{int}\,Score_{x,y,\theta,i}$ for every pixel.

Firstly, every element in **pointscore** is labeled 0 if the corresponding element in **m** is an occupied pixel. Otherwise element is labeled infinity. For *i* from 0 to the end, an infinity element, which is 4-connected with an element labeled *i,* should be changed into *i*+1. The pseudocode is shown in Fig.10. It is a Breadth-First Search (BFS) process. The **pointscore** can be addressed by a queue, which is time efficient.

```
every element in pointscore is labeled infinity
for the pixel in m is occupied do
    the corresponding element in pointscore is labeled 0;
    push this element into queue;
end for
while queue is not empty do
    for the element 4-connected with the HEAD do
        if the element is infinity then
            this element is labeled element_HEAD+1;
            push this element into queue;
        end then
    end for
    dequeue;
end while
```

Fig.10. a pre-calculated dealing with **pointscore**

As can be seen in Fig.11, we use a simple map to indicate the **pointscore** and the Manhattan Distance from a pixel to a nearest occupied pixel. To see clearly, 0 is replaced by asterisk.



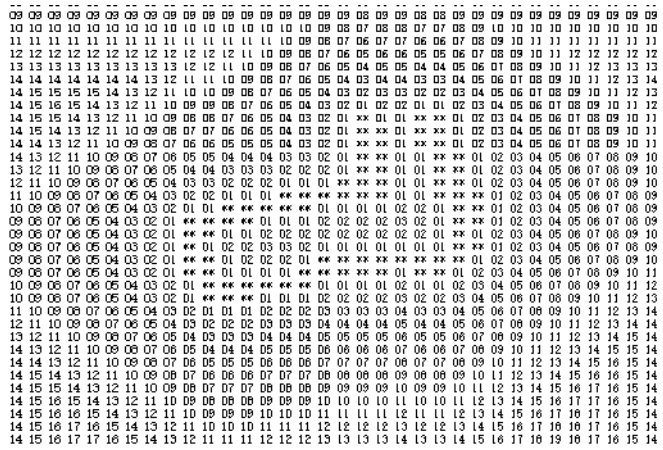


Fig.11. an example of a simple map and its **pointscore**

After we figure out the **pointscore**, every $Po\,\mathrm{int}\,Score_{x,y,\theta,i}$ can be inquired in O (1).

IV. EXPERIMENT

We tested the accuracy and the time cost of the novel global localization system in the laboratory which is a 100-square-meter room. Given a smaller area shown in Fig.12, in the system, the lowest scored pixel is (64,188) and with the orientation 165 degree. And the static map matching stage takes 3 seconds to figure out the initial pose.

Fig.12. an example of static map matching method

Even in the kidnapped problem, shown in Fig.13, we picked up the robot off the ground, and located it at another position. Then the cliff sensors noticed that situation and stopped the process. Once the robot get back to the ground, the global system restarted with calculating the new initial pose. It also cost 3 seconds.

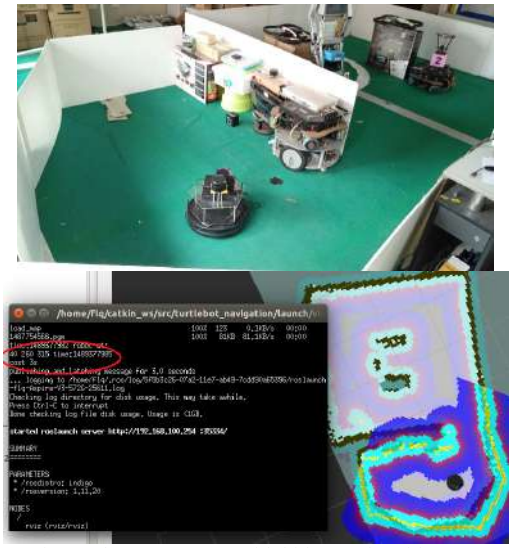


Fig.13. the system cope with kidnapped problem.

For the full map of the laboratory, shown in Fig.14, it has symmetrical and similar area, which is difficulty in localization. The robot was located at left and right corridor, and for the bigger space, the system spent about 9 seconds in processing. Both tests are accepted, shown in Fig.15 and Fig.16.

Fig.14. full map of a 100-square-meter laboratory

Fig.15. the robot was located at left corridor

Fig.16. the robot was located at right corridor

V. CONCLUSION

In conclusion, the novel global localization system is divided into two stages. In first and static stage, when the robot boots up in navigation mode within a known 2D map of environment, the map matching approach would pre-compute the **pointscore**, and then combine the map pixels into grids that consist of 4 × 4 pixels. The system assumes that every grid as well as 24 evenly spaced orientation might be the initial pose of the robot in the real world and calculate the $PoseScore(x, y, \theta)$ respectively. Subsequently, the system chooses the lowest scored $pose(x, y, \theta)$ as the initial pose and publishes it to ROS. This enumerated pose in the map represents an approximate pose in the real world at which the robot is located. In second and dynamic stage, the system adopts the Adaptive Monte Carlo algorithm with particle filter in ROS to correct and track the moving robot's pose in real time. Due to the approximate pose we set in the first stage, the particle would converge properly. It realizes that the system figures out the robot's initial pose in the startup stage by itself without motion and the robot can start to navigation process directly at the beginning. In practice, it is a feasible system to fix the global localization problem. For users, less than 10 seconds in startup stage is tolerable.

ACKNOWLEDGMENT

This research work is supported by Guangdong province science and technology plan projects (2015A020219001, 2017A010101031). The Fundamental Research Funds for the Central Universities (2015ZM140, 2017MS048). Guangzhou Key Laboratory of Robotics and Intelligent Software (15180007). Shenzhen peacock project (KQTD20140630154026047). Shenzhen basic research projects(JCYJ20160429161539298). Guangdong Ministry of Education Foundation (2013B090500093).

REFERENCES

[1] H. Ahmad and T. Namerikawa, "Robot localization and mapping problem with bounded noise uncertainties," *2012 IEEE Symposium on Industrial Electronics and Applications*, Bandung, 2012, pp. 187-192.

[2] Fengbing Luo, Bianjing Du and Zhen Fan, "Mobile robot localization based on particle filter," *Proceeding of the 11th World Congress on Intelligent Control and Automation*, Shenyang, 2014, pp. 3089-3093.

[3] L. Zhang, R. Zapata and P. Lépinay, "Self-adaptive Monte Carlo localization for mobile robots using range sensors," *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*, St. Louis, MO, 2009, pp. 1541-1546.

[4] Y. Abdelrasoul, A. B. S. H. Saman and P. Sebastian, "A quantitative study of tuning ROS gmapping parameters and their effect on performing indoor 2D SLAM," *2016 2nd IEEE International Symposium on Robotics and Manufacturing Automation (ROMA)*, Ipoh, 2016, pp. 1-6.

[5] J. Xiong, Y. Liu, X. Ye, L. Han, H. Qian and Y. Xu, "A hybrid lidar-based indoor navigation system enhanced by ceiling visual codes for mobile robots," *2016 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, Qingdao, 2016, pp. 1715-1720.

[6] Yi-Chun Lin, Chih-Chung Chou and Feng-Li Lian, "Indoor robot navigation based on DWA*: Velocity space approach with region analysis," *2009 ICCAS-SICE*, Fukuoka, 2009, pp. 700-705.

[7] D. Lee, S. Lim, Y. Lee and J. Yoon, "Map refinement for mobile robot navigation by polygonal approximation and distance transformation," *2016 10th International Conference on Signal Processing and Communication Systems (ICSPCS)*, Gold Coast, QLD, 2016, pp. 1-3.

[8] L. P. N. Matias, T. C. Santos, D. F. Wolf and J. R. Souza, "Path Planning and Autonomous Navigation using AMCL and AD*," *2015 12th Latin American Robotics Symposium and 2015 3rd Brazilian Symposium on Robotics (LARS-SBR)*, Uberlandia, 2015, pp. 320-324.

[9] Y. C. Lee, W. Yu, J. H. Lim, W. K. Chung and D. W. Cho, "Sonar Grid Map Based Localization for Autonomous Mobile Robots," *2008 IEEE/ASME International Conference on Mechtronic and Embedded Systems and Applications*, Beijing, 2008, pp. 558-563.

[10] Y. Wang, D. Wu, S. Seifzadeh and J. Chen, "A moving grid cell based MCL algorithm for mobile robot localization," *2009 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, Guilin, 2009, pp. 2445-2450.