A Realistic Simulator for Humanoid Soccer Robot Using Particle Filter

Yao Fu, Hamid Moballegh, Raúl Rojas, Longxu Jin and Miao Wang

Abstract This work presents a realistic simulator called Reality Sim for humanoid soccer robots especially in simulation of computer vision. As virtual training, testing and evaluating environment, simulation platforms have become one significant component in Soccer Robot projects. Nevertheless, the simulated environment in a simulation platform usually has a big gap with the realistic world. In order to solve this issue, we demonstrate a more realistic simulation system which is called Reality Sim with numerous real images. With this system, the computer vision code could be easily tested on the simulation platform. For this purpose, an image database with a large quantity of images recorded in various

Based on <Reality Sim: a realistic environment for robot simulation platform of humanoid robot>, by <Yao Fu, Hamid Moballegh, Raúl Rojas, Longxu Jin, Miao Wang> which appeared in the proceedings of the 5th International Conference on Automation, Robotics and Applications (ICARA2011).

Y. Fu (✉) · L. Jin
ChangChun Institute of Optics, Fine Mechanics and Physics,
Chinese Academe of Sciences, ChangChun, China
e-mail: yao.fu.felicity@gmail.com

L. Jin
e-mail: jinlx@ciomp.ac.cn

Y. Fu
Graduate University of Chinese Academe of Sciences, Beijing, China

Y. Fu · H. Moballegh · R. Rojas · M. Wang
Department of Mathematics and Computer Science, Freie Universität Berlin,
14195 Berlin, Germany

H. Moballegh
e-mail: moballegh@gmail.com

R. Rojas
e-mail: Rual.Rojas@fu-berlin.de

M. Wang
e-mail: miao.wang@fu-berlin.de

G. Sen Gupta et al. (eds.), *Recent Advances in Robotics and Automation*,
Studies in Computational Intelligence 480, DOI: 10.1007/978-3-642-37387-9_21,

camera poses is built. Furthermore, if the camera pose of an image is not included in the database, an interpolation algorithm is used to reconstruct a brand-new realistic image of that pose such that a realistic image could be provided on every robot camera pose. Systematic empirical results illustrate the efficiency of the approach while it effectively simulates a more realistic environment for simulation so that it satisfies the requirement of humanoid soccer robot projects.

1 Introduction

The research in this paper is based on the humanoid league of soccer competition in RoboCup [1]. As an international robotics competition, RoboCup aims to foster AI and intelligent robotics research by providing a standard problem where a wide range of technologies can be integrated and examined.

A soccer robot project is comprised of many sub-systems, e.g., vision, planning and motion. Simulation platforms [2, 3] are contributive in developing, testing and improving Control, Computer Vision and other relevant aspects in robot soccer competition. Nevertheless, the simulation environment and the real world have large difference which attracts many researchers to put effort to lessen it. Jakobi, N [4] proposed minimal simulations for evolutionary robotics within which controllers that meet these conditions will successfully transfer into reality. Tom Ziemke [5] discussed significance of a robot simulator in experimentation with active adaptation of non-trivial environments. Juan Cristóbal Zagal et al. [6–8] developed a simulator which is called UCHILSIM for RoboCup four-legged league, aiming at solving the so-called reality gap. Josh C. Bongard et al. [9]. presented a new co-evolutionary approach, by which algorithm, they automatically adapt the robot simulator using behavior of the target robot and adapt the behavior of the robot using the robot simulator. In literature [10], some more methods are proposed to assess discrepancies between the simulation and the reality.

Nevertheless, the gap between reality and simulation platform still exists. The papers mentioned above mostly focus on the controlling function between them. Actually, the gap in computer vision aspect is obviously not realistic enough, i.e. the testing environment is commonly a simulated image, but not the real image. In this chapter, we develop a novel subsystem, which is called Reality Sim for reducing the gap between the reality and simulation especially in vision aspects.

First, an image database was built with huge mount of real images. The images are captured by the robot from FUmanoids Team [11]. The team attended the Humanoid league of RoboCup champion each year from 2006 and had excellent records of two times 2nd place and one time 3rd place. Secondly, in order to record the images by three-dimension camera poses, the Particle Filter method is applied for recording camera poses of the images. The camera pose recording problem is similar to the well-known robot self-localization problem. Thirdly, although

thousands of images recorded by camera pose are in the database, it is hardly possible to incorporate all camera poses. Every camera pose includes three dimension variables-x, y coordinates and yaw angle hence it is too hard to incorporate so many three dimension points on a 400 × 600 cm^2 size soccer field just by building a database. Once a camera pose is not included in the database, an interpolation method is brought to reconstruct a brand-new image. By these approaches, the realistic image at every robot pose could be found from the database or retrieved by an image interpolation algorithm. Consequently, Reality Sim is of great use especially in developing and testing vision algorithms. Likewise, the simulator is more approximate to the realistic world.

The remaining of the paper is organized as follows, the camera pose recording method of images is explained in the second section, among which, the Particle Filter algorithm is adopted. The third section introduces image interpolation process. After that, experiments are shown in the fourth section, followed by conclusions drawn finally.

2 Camera Pose Calculation

In the Reality Sim subsystem, with the purpose of returning an image at designated pose to simulator, the camera pose was recorded previously. Camera pose calculation is similar with conventional mobile robot self-localization [12] in soccer robot scenario. For Self-localization problem, the robot's pose which generally includes location and orientation relating the environment is estimated from sensor data. In a humanoid robot system, Self-localization is a crucial problem since many other modules e.g., world model, behavior control depends on it. The camera pose computation could be regarded as a robot self-localization problem.

2.1 Particle Filter Algorithm

Many strategies have been applied to robot self-localization issue, e.g., Kalman Filter [13], grid-based method [14], Multiple Model Kalman Filters [15] and Particle Filter and so on. Among these, Particle Filter is the most popular approach due to its excellent performance in solving robot global localization problem. As the so-called Sequential Monte Carlo (SMC) method [16–18], Particle Filter represents the estimation the state of a dynamic system by a set of "particles", which are differently-weighted samples of the state space.

The Sequential Monte Carlo (SMC) method is mainly composed of motion, observation, resampling and state estimation. The steps of camera pose calculation, or rather, self-localization method in this paper is depicted as follows,

- *Initialization with N samples*: In the initialization, the weight of every sample is assigned a same initial value. One sample includes three variables, x, y coordinates and yaw angle.
- *Prediction*: As the so-called motion model, in prediction step, the "predictive" particles are got to predict the current position. Suppose x_t is state (location) at time t, y_t is sensor readings at time t, u_{t-1} is control command, like action, velocity and so on at time $t-1$. u_{t-1} is used to predict the position at time t. In this chapter, the samples generated from previous frame camera pose are introduced to predict the current position at t.
- *Observation model* (*Update*): In the robot soccer scenario, the observation model calculates what the robot observes when it is at specific location by the sensor. The observation model $P_t(y_t|x_t)$ is used to update the posterior belief $p_t(x_t)$ in each time step. The belief $p_t(x_t)$ is represented by some weighted samples. The evaluation of the weight ω_t^i ($i = 1$–N) will be explained in section B.
- *Resampling*: Resampling aims at eliminating samples that have small weights and concentrating on the ones with large weights thereby reducing effect of degeneracy problem. The posterior belief $p_t(x_t)$ is resampled for N_s times, afterwards, a new sample set $\{\tilde{x}^i\}_{i=1}^N$ is got and $\omega_i^k = 1/N_s$.
- *State Estimating*: Many measures e.g., overall averaging, best particle, etc., could be used in estimating state [19]. In other words, computing a robot pose via given sample set. In this chapter we select winner-take-all strategy, i.e., best particle method.

Figure 1 illustrates the camera pose calculation process via particle filter method for one image frame. T is the maximum iteration time. The iteration begins with N initialized particles. In each iteration, each sample is given a weight according to the evaluation method which will be mentioned below and the resampling is afterwards adopted. With these steps, the particles gradually focus on the most probable states.

2.2 Observation Model

In the whole process, the observation model is essential to estimate the weight of samples. In this chapter, we separately put forward two models, i.e. distance-based model and Gauss-based model for seeking a more efficient model.

2.2.1 Distance-Based Model

In this model, weight of every sample is updated by an evaluation function. Further more, the evaluating process mainly includes three steps. Firstly, Projecting the field lines onto the image by camera model [20] from vision subsystem of

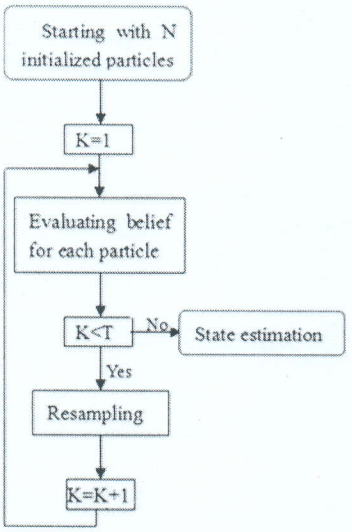


Fig. 1 The chart of camera pose calculating process

FUmanoids project according to its absolute coordinate. The projection green lines which are projected onto the real field white lines are shown in Fig. 2. This step comprises a translation from absolute coordinate in real world to image coordinate on the image via the robot pose. Secondly, calculating the distance between projection of field line and the real field line on the image by color feature in two opposite directions. The distance comparing process is also shown in Fig. 2. The short green and blue lines direct from the projection to the real field line are in two opposite directions. As a soccer playing field is a relatively simple environment, not too much feature could be utilized; as a result, the white line on the field is one of the most obvious features in this environment. In the meanwhile, contour of the field is found to eliminate effect of irrelevant data outside the field. Finally, as is shown in Fig. 2, for a projection line, there are many line-pairs from the projection to real field lines, selecting a shorter one of the line pair and saving its distance value. The sum of distance values are used to calculate the weight of each sample. The weighting method is shown in Eq. (1) as follows,

$$\omega_i = e^{\sum_{j=1}^{M} \frac{Dis_{ij}}{\alpha}}, i \in (1, N) \tag{1}$$

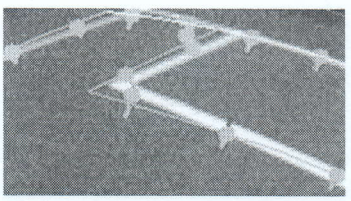


Fig. 2 The distance between the projection and real field line

where ω_i is the weight of the ith sample, N is sample number, Dis_{ij} is the jth distance of the ith sample, M is the distance number and α is a factor for adjusting range of weight. Afterwards, weights of samples are normalized so that the sum of weights equals 1.

2.2.2 Gauss-Based Model

Another observation model proposed in this chapter is Gauss-based model. Although the distance-based model is effective in calculating the weight value, it is time-consuming when the smaller distance of the two orthorhombic directions is detected. For this reason, the Gauss-based observation model for the particle filter method is proposed which is more universal for every frame and faster than the previous model. The mainly steps of this model is shown as follows,

1. The original image is translated into a gray image. Then Gauss blurring method is used afterwards. This method could effectively reduce noise and camera artifacts. The image after Gauss blurring is shown in Fig. 3e.
2. Detecting border of the field is used to further decrease the irrelevant data. The red line around the border of the field in Fig. 3a is the detected border line. The border detection function is included in the computer vision subsystem of the FUmanoids [11] project.
3. Projecting the camera model line onto the image for every particle. The image after projection is shown in Fig. 3a, b–d demonstrate the detail information of red frames from the image in Fig. 3a in which the green projection lines are shown.
4. Calculating accumulated pixel values on the projected lines. The accumulated value is signed to every particle as the weighting value. The weighting formula is shown in Eq. (2),

$$\omega_i = e^{\sum_{j=1}^{M} \frac{Pix_{ij}}{\alpha}}, i \in (1, N) \tag{2}$$

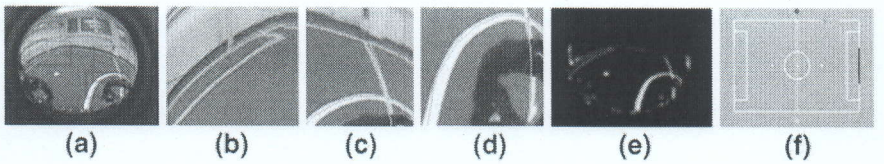


Fig. 3 The Gauss-based observation model of the particle filter algorithm. **a** is the image after projection; **b**, **c**, **d** shows the detail information in red frame of the image (**a**); **e** is its Gauss-transformed image; **f** shows the detected robot pose on the field by particle filter algorithm

in which ω_i is still the weight of the ith sample, N is sample number, Pix_{ij} is the jth pixel value on the image of the ith sample, M is the quantity of pixel points at which the pixel value is calculated. α is the factor for adjusting range of weight as Eq. (1). Afterwards, sum of sample weights is normalized to 1.

2.3 Resampling

The resampling step is essential to avoid the problem of degeneracy of the algorithm after several iterations and its basic idea to eliminate particles with small weights and concentrating on particles with large weights. The typical resampling algorithm namely Residual resampling [21] is adopted in this chapter. Considering a representation of current sample set $\{x_i^k, \omega_i^k\}$, k $= 1 \ldots N_s$, which represents the probability density function of current robot pose, a new set $\{x_i'^k, \omega_i'^k\}, k = 1 \ldots N_s$ is resampled so that $x_i^k = x_i'^l, k, l \in [1, N_s]$ and weight $\omega_i'^k = 1/N_s$.

3 Image Reconstruction

Although a great amount of images are stored in image database, it is yet impossible to comprise all camera poses. Since for the whole 400 $\times$ 600 cm^2 size soccer field and three dimensions pose-x, y coordinates and yaw angle, 720,000 of poses are needed. It is hard to traverse every camera pose in such a huge state space just by building an image database. Once the simulation platform calls for an image which is not included in the database, an image interpolation is invoked to interpolate a brand-new image and return it back. In mathematical field, Interpolation means a construction of new data points within the range of a discrete set of known data points. The data are obtained by sampling or experimentation. In this chapter, regression analysis [22] is adopted to interpolate a new image by its nearby images.

The reconstructed image will be interpolated by its nearest neighbor images. In order to find out the correlation of camera poses between the two images, a regression function is defined, i.e. a mathematic model is built. After that, unknown parameters in the model are estimated from sample data.

A regression model relating Z to a function of X, Y and β is shown as follows,

$$Z = f(X, Y, \beta) \tag{3}$$

Z is a dependent variable, X and Y are independent variables, β is unknown parameter matrix, f is model function. In order to estimate unknown parameters β, a solution to minimize the distance between the measured and predicted values of

the variable Y, Least-Squares [23, 24] is used. Once β is estimated, Y could be calculated by the model function.

According to this model, the function f for interpolating among images is given as follows,

$$\begin{bmatrix} X_2 \\ Y_2 \end{bmatrix} = \begin{bmatrix} \alpha_0 & \beta_0 \\ \alpha_1 & \beta_1 \\ \alpha_2 & \beta_2 \\ \alpha_3 & \beta_3 \\ \alpha_4 & \beta_4 \\ \alpha_5 & \beta_5 \end{bmatrix}^{\mathrm{T}} \begin{bmatrix} 1 \\ X_1 \\ Y_1 \\ X_1 Y_1 \\ X_1^2 \\ Y_1^2 \end{bmatrix} \tag{4}$$

As is shown in Eq. (4), (X_1, Y_1) are pixel points sampled from camera pose of nearby image. Whereas, (X_2, Y_2) are corresponding points on current robot pose. The α_i and β_i are respectively unknown parameters. Firstly, the corresponding sample point pair (X_1, Y_1) and (X_2, Y_2) are separately selected from the nearby image and image of current camera pose. After finding the relation between the image coordinate and absolute coordinate, the pixel values on image whose absolute coordinate value are the same are used as sample data. Secondly, Least squares method [22, 23] is adopted for estimating model parameters α_i and β_i subsequently. After the parameters are estimated, the model is built. Thirdly, (X_2, Y_2) points are calculated from the model and they are furthermore interpolated onto current robot image. Afterwards, the correlation between the two camera poses is found. Furthermore, if some pixels on image which belong to current robot pose do not have correlation with nearby image, it could not be assigned into any value. For this kind of pixel points, a reverse model, namely a model from current camera pose to its nearby image is built and the corresponding RGB values of pixels from nearby image is assigned to these pixels subsequently. The reconstructed images is shown in Fig. 4b.

Up to now, the main frame of the system is depicted. Previously, building an image database in which images are recorded by camera pose. Then when the simulation platform calls an image at some camera pose from Reality Sim, an image in the database or a reconstructed new image is returned.

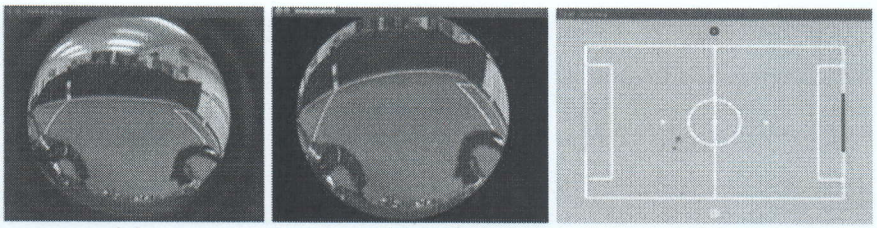
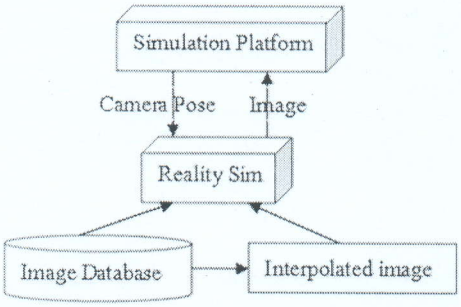


Fig. 4 **a** Is the interpolated image; **b** is its nearby image; **c** shows the robot pose on the field

Fig. 5 System chart



The system chart is shown in Fig. 5. This system could be efficiently utilized in testing the vision code on simulator.

4 Experimental Results

In the experiment, both of the two robot platforms-2009 and 2011 platform of FUmanoids soccer robot project which separately participated the RoboCup 2009 [11], 2010 [11] and RoboCup 2011 [25] are adopted. One robot in the FUmanoids team 2009 is shown in Fig. 6a and the image captured by this robot is shown in Fig. 6b. Figure 6c and d shows the robot-Anna in FUmanoids team 2011 and the captured image. Both of the captured images are 480 × 640 pixel size. The experiment environment is shown in Table 1. Figure 7 is the simulation platform of FUmanoids project.

An image database is built at first. The robot soccer field [26] which is 400 × 600 cm^2 size is shown in Fig. 8. The images are captured by a fish-eye camera equipped on the robot which covers a full range of 180 × 90°. For building database, a robot from FUmanoids soccer robot team moved successively and randomly on the soccer field. In the meanwhile, are captured images. To measure the validity of our approach, a ground truth database, the images in which are captured every 30 cm interval and manually recorded by camera pose is utilized as ground truth.

In the database about 10,000 images are indexed by camera pose. They are tagged by pose information, x, y coordinates, yaw, pitch and roll angle. As is depicted in part II, Particle filter algorithm is putted to use for tagging the images by robot pose.

Figure 9 illustrates the process of particle number selection for particle filter algorithm. The accuracy of the method increases with the number of samples. However, increased particle quantity causes more processing time. While the particle number is up to 1,000, the varying of the average error is not obvious. Considering time consuming and precision, sample number is set to 800. For this experiment, the Gauss-based observation model is adopted.

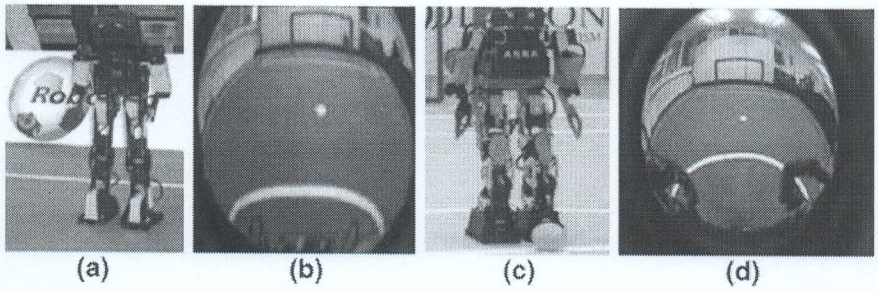


Fig. 6 **a** Robot-Eve of FUmanoids 2009 platform; **b** The image captured by Eve; **c** robot-Anna of FUmanoids 2011 platform; **d** The image captured by Anna

Table 1 Experimental environments

Developing environment	Parameters
PC configuration	Intel i5 2.5 GHz, RAM 4 GB
Operation system	Ubuntu 10.10
Programming language	C++ & OpenCV 1.0
Development environment	Eclipse Helios 3.6

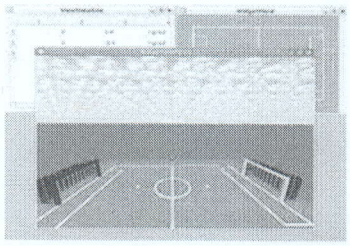


Fig. 7 The simulation platform of FUmanoids project

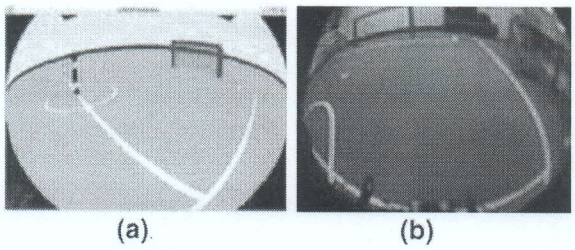


Fig. 8 A result of camera pose recording

To evaluate the effects of two observation models, the corresponding experiment results are shown in Table 2. The Gauss-based observation model shows a smaller average error and obviously shorter computation time. For this experiment, the particle number equals 800. In the mean while, $-1{,}000$ is assigned to factor α in formula (1) and (2). The initial values of the samples are set according to pose of the first frame of an image sequence.

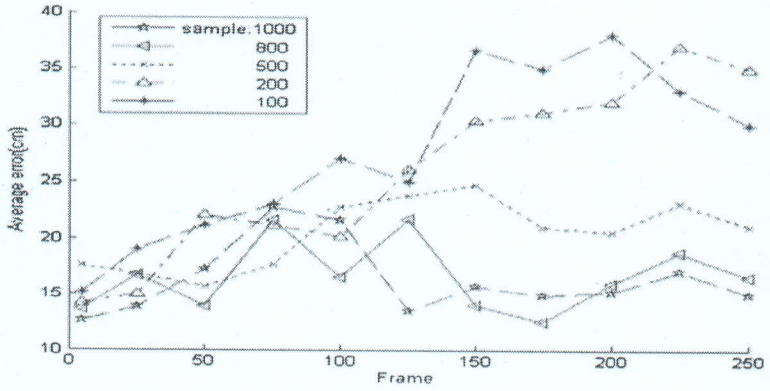
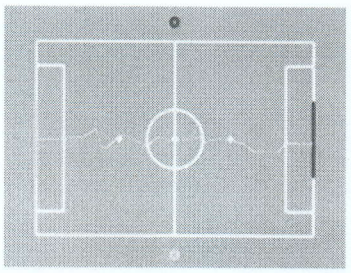


Fig. 9 Selection of particle number

Table 2 Comparison of two likelihood measures

Method	Average error (cm)	Average time (s)
Distance-based	18.5	1.76
Gauss-based	18.1	0.85

Fig. 10 **a** A sample of the simulated image; **b** A sample of the interpolated image



Another experiment for evaluating recording method by particle filter algorithm, a robot moving route is drawn in Fig. 10. The robot moves from the middle of one side of the field to the other side of the field straightly just like the route of ground truth data. Contrasting with ground truth data, the average distance error is 18.51 cm. An essential aspect which causes the error is that if lines are very far away from the robot, it is hard to recognize the color of the line exactly.

The following experiment is relevant to image interpolation. Figure 8 shows the comparison of the simulated image and the interpolated image. Compared with the real image, the interpolated image is obviously more realistic.

The point-pair (X, Y) for interpolation is selected by scanning the whole soccer field at 25 cm $\times$ 25 cm intervals. In this experiment, a route of robot pose is selected as is shown in Fig. 11. In the experiment, the robot moves from (0, 3, 0) to (8, 6, 4) continuously. The variables of 3-D pose respectively stands for x, y coordinates and yaw angle. The images at every robot pose are shown in following Fig. 12.

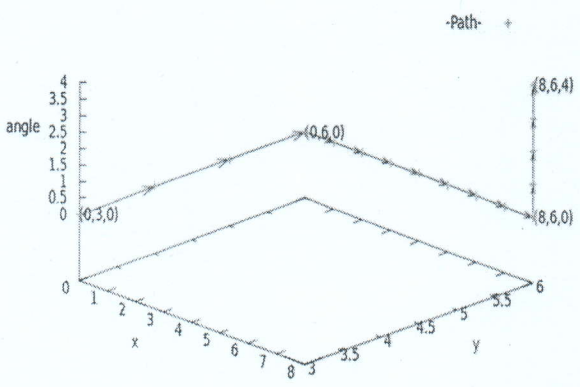


Fig. 11 The path of robot pose

Fig. 12 An image sequence when the robot poses changes successively from (0, 3, 0) to (8, 6, 4). (the sequence is from left to right, images are from the FUmanoids platform 2009.)

As is shown in Fig. 12, when robot pose moving continuous from (0, 3, 0) to (8, 6, 4), the image at current pose is returned to the system. By images interpolation, the images vary continuously and smoothly. The above experiments demonstrate the whole system provides a more realistic world for the simulation platform.

5 Conclusion

In this chapter we presented a simulation subsystem-Reality Sim which is utilized mainly in testing vision code in simulation platform of a Humanoid soccer robot project. In this system, great quantities of real images are captured by the robot camera and stored in an image database. Besides recording the camera pose by camera model and Particle Filter method, an image interpolation method is adopted to get the images which are not included in database. With this system when a robot moves smoothly on the simulated soccer field, an image at current pose will be returned immediately. This work efficiently satisfies the need of vision code testing and extends the application area of simulation platform. In the future, we will enrich this system to make it even more realistic.

Acknowledgments The authors gratefully acknowledge Daniel Seifert for his knowledge of the project and other members of FUmanoid Team for providing the software base for this work. A video which is relevant to the chapter is linked: http://www.youtube.com/watch?v=TjjBYVMxZak.

References

1. H. Kitano et al. RoboCup: a challenge problem for AI and robotics. RoboCup-97: Robot Soccer World Cup I, (Springer, Heidelberg 1998), pp. 1–19
2. K. Asanuma, K. Umeda, R. Ueda, T. Arai, in *Development of a Simulator of Environment and Measurement for Autonomous Mobile Robots Considering Camera Characteristics.* Proceedings of robot soccer world cup VII (Springer, Heidelberg, 2003)
3. T. Ishimura, T. Kato, K. Oda, T. Ohashi, in *An Open Robot Simulator Environment.* Proceedings of robot soccer world cup VII (Springer, Heidelberg, 2003)
4. N. Jakobi, Minimal simulations for evolutionary robotics. PhD thesis, University of Sussex, 1998
5. Ziemke, On the role of robot simulations in embodied cognitive science. AISB J. **1**(4), 389–399 (2003)
6. M. Young, The Technical Writer's Handbook. Mill Valley, CA: Juan Cristobal Zagal and Javier Ruiz-del-Solar. Combining simulation and reality in evolutionary robotics. J. Intell. Robot Syst. **50**(1), 19–39 (2007)
7. J.C. Zagal, J. Ruiz-del-Solar, in *UCHILSIM: A Dynamically and Visually Realistic Simulator for the RoboCup Four Legged League*, vol. 3276 . RoboCup 2004: Robot soccer world cup VII, lecture notes in computer science (Springer, Berlin, 2004), pp. 34–45
8. J.C. Zagal, J. Ruiz-del-Solar, P. Vallejos, in *Back-to-Reality: Crossing the Reality Gap in Evolutionary Robotics.* IAV 2004: Proceedings 5th IFAC symposium on intelligent autonomous Vehicles, Elsevier Science Publishers B.V. AISB J. 1(4), 389–399 (2004)
9. J.C. Bongard, H. Lipson, in *Once More Unto the Breach: Co–Evolving a Robot and its Simulator.* Proceedings of the ninth international conference on the simulation and synthesis of living systems (ALIFE9), pp. 57–62
10. L. Iocchi, F. Dalla Libera, E. Menegatti, in *Learning Humanoid Soccer Actions Interleaving Simulated and Real Data.* Proceedings of the second workshop on humanoid soccer robots IEEE-RAS 7th international conference on humanoid robots, Pittsburgh, 2007
11. B Fischer et al. FUmanoid team description paper 2010. (Workshop Robocup Singapore 2010)
12. S. Thrun, D. Fox, W. Burgard, F. Dellaert. Robust Monte Carlo localization for mobile robots. Artif. Intell. **128**(1–2), 99-141 (2001)
13. R.E. Kalman, A new approach to linear filtering and predictionproblems. J. Basic Eng. **82**(1), 35–45 (1960)
14. M.J. Quinlan, R.H. Middleton, in *Comparison of Estimation Techniques Using Kalman Filter and Grid-Based Filter for Linear and Non-Linear System.* Proceedings of the international conference on computing: Theory and applications technique for RoboCup soccer (*ICCTA2007*) (1960)
15. M.J. Quinlan, R.H. Middleton. Multiple model kalman filters: a localization technique for RoboCup soccer. Lect. Notes Comput. Sci. **5949**, 276–287 (2010)
16. S. Thrun, W. Burgard, D. Fox, *Probabilistic Robotics* (MIT Press, Cambridge, 2005)
17. A.De Doucet, N. Freitas, N.J. Gordon, *Sequential Monte Carlo Methods in Practice* (Springer, Heidelberg, 2001)
18. M.S. Arulampalam, S. Maskell, N. Gordon, and T. Clapp, A tutorial on particle filters for on line nonlinear/non-gaussian bayesian tracking. IEEE Trans. Sig. Process. **50**(2) (2002)
19. T. Laue, T. Röfer, in *Pose Extraction from Sample Sets in Robot Self-localization-a Comparison and a Novel Approach.* Proceedings of the 4th European conference on mobile robots—ECMR'09, (Mlini/Dubrovnik, Croatia, 2009), pp. 283–288
20. T. Langner, Selbstlokalisierung für humanoide Fußballroboter mittels Mono-und Stereovision. Master thesis. FU Berlin, FB Mathematik und Informatik, Berlin. September 2009 (in German)

21. R. Douc, O. Cappe, E. Moulines, in *Comparison of Resampling Schemes for Particle Filtering*. ISPA 2005. Proceedings of the 4th international symposium on image and signal processing and analysis (2005), pp. 64–69
22. A. Desrosières, *The Politics of Large Numbers: a History of Statistical Reasoning*, Trans. Camille Naish (Harvard University Press, United State, 2004)
23. A. Björck, *Numerical Methods for Least Squares Problems* (SIAM, Philadelphia, 1996)
24. J. Nocedal, J. Stephen, *Wright Numerical Optimization* (Springer, Heidelberg, 1999)
25. D. Serfert et al. FUmanoid team description paper 2011. Workshop RoboCup Istanbul (2011)
26. RoboCup soccer humanid league rules and setup, http://www.tzi.de/humanoid/bin/view/Website/Downloads