Proceedings of the 13th Automation & Computing Society Conference in the UK, Stafford, England, 15 September 2007

Sensor-Based SLAM for Camera Tracking in Virtual Studio Environment

Po.Yang, Wenyan.Wu, M.Moniri and Claude C. Chibelushi
Faculty of Computing, Engineering and Technology
Staffordshire University
Stafford, UK
P.Yang@staffs.ac.uk, W.Wu@staffs.ac.uk

Abstract— **This paper addresses the problem of Camera Tracking in virtual studio environment. The traditional camera tracking methods are vision-based or sensor-based. However, the Chroma Keying process in virtual studio requires the color cues, such as blue screen, to segment objects from images and videos. It limits the application of vision-based tracking methods in virtual studio since the background could not provide enough feature information. Therefore, in our research, we would try to apply the SLAM (simultaneously localization and mapping) methodology from mobile robots to the camera tracking area. We describe a sensor-based SLAM extension algorithm for 2D camera tracking in virtual studio. Also a technique call Map Adjustment is proposed to increase the accuracy and efficiency of the algorithm. The simulation results would be given in the conclusion.**

Keywords-SLAM, Particle Filter, Chroma Keying, Camera Tracking

I. INTRODUCTION

Camera tracking in unprepared scenes is a hot research topic recently since it could be applied into various areas, etc augmented reality, virtual reality. In virtual studio environment, it is difficult to obtain a system, which is sufficiently accurate, fast and robust for effective camera tracking and also suitable for chromakeying process in video or movie production. The chromakeying is the process of segmenting objects from images and video using color cues. A blue screen placed behind an object during recording is used in special effects and in virtual studios. The blue color is later replaced by a difference background.

The existing camera tracking techniques are classified into vision-based tracking methods and methods based on sensors tracking. The vision-based tracking approaches [1] [2] are based on image information, they track position and rotation of a camera by using the information contained in images sequences, such as fiducial marker or feature points. Though SLAM have been applied into the pure vision-based tracking method, the high input data rate, the inherent 3D quality of visual data and the difficulty in extracting long-term features to map limit the range of its possible applications of vision-based tracking system.

The sensor-based tracking methods are based on active sensors, which incorporate powered signal emitters and sensors placed in a prepared and calibrated environment, such as magnetic [3], optical[4], radio, and ultrasound-guided. However, most of the active (sensor-emitter) tracking systems directly observe the position or orientation parameters of camera so that it is easy to be inaccurate due to the drawback of sensor system. Magnetic tracking suffers in terms of jitter; optical tracking is computationally expensive and slow.

In the last several decades, SLAM technology has been of great interest for mobile computing and robotic researchers. In the field of mobile computing, by providing the location information of a user, it can be applied to build others application. In order to know the information about robot environment, the sensor measurements deliver information about the bearing, distance, appearance etc. of nearby features in the environment. There have been various mobile computing systems providing in-door localization, using sensors like: Ultrasonic, Infrared, Laser or Radio frequency. Some of these systems are commercial and have achieved impressive success, e.g. the Active Badge System [5] by AT&T lab in Cambridge University, the RF transmitter and ultrasonic transmitter system in Bristol University by Randel and Muller [6].

In our research for virtual studio, the chromakeying process limits the application of vision-based methods in virtual studio since the blue screen would not give enough feature information for tracking. So our work would is highly focused on the application of the SLAM methodology from mobile robot to the camera tracking area. Since the SLAM methodology does not depend on the individual character of sensor system, it would use the statistic and probabilistic algorithms to track the camera by only using simple

In this paper, Section 2 gives a basic literature review of virtual studio and SLAM methodology. Section 3 presents sensor-based SALM algorithm for camera tracking. Section 4 describes the simulation results and analysis. Section 5 draws some conclusions and future work.

II. LITERATURE REVIEW

A. *Virtual Studio*

Virtual studios have long been in use for commercial broadcasting and motion pictures. Most of virtual studios are based on “blue screen” technology, and its two-dimensional (2-D) nature restricts the user from making natural three dimensional (3-D) interactions. In general, virtual studio sets require “blue screen” (Chroma Keying) technology, high-end graphics workstations, camera tracking technology and signal compositing for high realism, and exact mixing results [7]. The 2-D nature of current virtual studios as developed and used in the current broadcast industry limits its use to situations where the camera angle is fixed and there is minimal user interaction [8].

Yamanouchi overcome the limitation of the cost and space for the conventional blue-screen setups in their "real space-based virtual studio" system [9]. This system combines the real and virtual space images. The real and virtual images are mixed using the depth information in real time, but their algorithms are limited only to indoor studio sets. The most straightforward method of extracting 3-D information from a scene is to use multiple camera views and stereo correspondences. Scharstein and Szeliski gave a very good survey and taxonomy of the different algorithms and approaches to the stereo-correspondence problem [10]. Yang introduced a new method for using commodity graphics hardware to achieve real-time 3-D depth estimation by a plane-sweeping approach with multiresolution color-consistency tests [11]. These methods are the main approaches for the extraction of camera parameters.

Meanwhile, Chroma Keying is a very important issue in virtual studio. Chroma Keying is used in video and movie production for replacing the background in special effects and in virtual studios applications and for hiding objects. It is a staple of video production, provides a good starting point for understanding the historical development of virtual studios. In traditional chromakeying, the subject is shot against a constant background such as a blue curtain or screen. This "blue screen " shot then passes through a chromakeyer, where it is combined with a second shot containing the new background. Conceptually, chromakeyer operation is simple: replace the foreground with the background in those places where the foreground contains a particular color known as the key color. The chromakeyer itself may do so automatically. Despite of chromakey systems' sophistication, their operation imposes a fundamental constraint: The foreground camera can not move- it must be "lock off" for the shot's duration. The spatial relationships existing between the two layers are not consistently maintained.

B. SLAM methodology

SLAM stands for Simultaneous Localization and Mapping, which has been a fundamental topic in robotic and mobile computing communities. In our research work, we would extend the SLAM method from mobile robot to camera tracking. Since the existing techniques in the literature are mostly concerning about robot, the description of SLAM methodology is more based on robotics. SLAM stands for Simultaneously Localization and Mapping, and Localization deals with the problem of trying to find the location of the robot, given a map and some sensor reading data. Mapping is the process of building and maintaining a model of the surrounded environments. Localization and mapping have been under active development during the past several decades, the first paradigm is called model-based [12] in 1970s. In the 1980's, the Brook's [13] behavior-based architecture becomes more popular. The last paradigm emerged since mid 1990s, which is still under rapid developing, is usually termed probabilistic robotics [14]. This method describes all the information in a probabilistic way, unlike the above methods which are deterministic. Recently, some vision researchers investigated SLAM algorithms [15] [16] in pure vision domain, however the vision-only SLAM systems suffer from the inherent 3D quality of visual data and the difficulty in extracting long-term features to map [17]. Thus, in this paper, the particle filter SLAM method is also based on the probabilistic techniques, but based on sensor instead of vision-based features.

In a probabilistic camera tracking system, the aim of localization is to estimate the state of the camera and its environment, from some sensor measurements. So we need a mathematical representation which can help to represent and calculate the estimations. Bayes filtering [18] address such a problem.

Bayes Theroem: Let $B_1,...B_k$ from a partition of space S such that $\Pr(B_i) > 0$, for $j = 1, \dots, k$, and A is an event such that $\Pr(A) > 0$. Then, for $i = 1, \dots, k$:

$$\Pr(B_i \mid A) = \frac{\Pr(B_i)\Pr(A \mid B)}{\sum_{j=1}^{k} \Pr(B_j)\Pr(A \mid B_j)} \qquad (1)$$

Where the nominator

$$\sum_{j=1}^{k} \Pr(B_j)\Pr(A \mid B_j) = \Pr(A)$$

The key idea of Bayes filtering is to estimate a probability density over the state space conditional on the given sensor data. It is often called Belief. If denote the state of the robot at time t by s_t, and the sensor data from 0 to t by $d_{0:t}$. The Belief of state s at time t can be written as:

$$Bel(s_t) = \Pr(s_t \mid d_{0:t}) \qquad (2)$$

Apply Bayes theorem (1) and Equation (2):

$$\begin{aligned} Bel(s_t) &= \Pr(s_t \mid d_t, d_{0:t-1}) \\ &= \eta \Pr(d_t \mid s_t, d_{0:t-1}) \Pr(s_t \mid d_{0:t-1}) \end{aligned} \qquad (3)$$

Where $\eta = \Pr(d_t \mid d_{0:t-1})^{-1}$ is a normalizing constant relative to s_t. (η is determined by the observation model and system noise). After applying Markov assumption, Equation (3) becomes:

$$\begin{aligned} Bel(s_t) &= \eta \Pr(d_t \mid s_t) \Pr(s_t \mid d_{0:t-1}) \\ &= \eta \Pr(d_t \mid s_t) \int \Pr(s_t \mid s_{t-1}) \Pr(s_{t-1} \mid d_{0:t-1}) ds_{t-1} \\ &= \eta \Pr(d_t \mid s_t) \int \Pr(s_t \mid s_{t-1}) Bel(s_{t-1}) ds_{t-1} \end{aligned} \qquad (4)$$

The Bayesian filter is used in most probabilistic localization system. It is, however, only a theoretical framework for this estimation problem. The integration in Equation (4) is a vital problem. If the state space is continuous, the implementation of Equation (4) requires memory storage for the representation of the whole posterior distribution, which is an infinite dimensional vector. In cases where the state space is discrete and of high dimensionality, the integration is still extremely complicated and not practical to implement.

III. Sensor-based SLAM algorithm

The aim of this algorithm is to achieve the sensor based 2D camera tracking in a virtual studio environment. This section provides a comprehensive description of the implementation of system states, system models and the particle filter in this algorithm. The particle filter in this SLAM algorithm is not exactly the same as the standard particle filter. In addition, the algorithm also has potential to be applied in different sensor network environment. The sensor-based SLAM for camera tracking could be as in Figure 1:

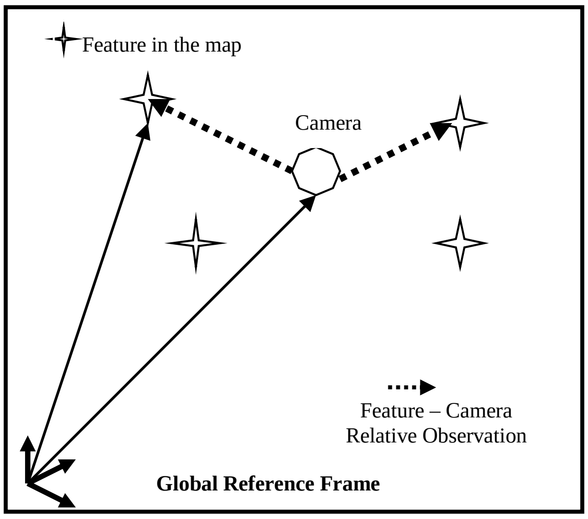


Figure 1 the SLAM for camera

A. *Systen state and model*

In this research work, it is assumed that the observation system is based on a sensor network, to successfully obtain the range information. Thus we just assume that there are several sensor transmitters mounted in the surroundings, and the camera is equipped with a sensor receiver. Actually, the system just only requires the distance between the camera point to the feature points over time. Hence, the sensor network could comprise active sensor or passive tags. Then each feature of the map actually represents a node of sensor transmitter, they are denoted as f_n , where n is an index of transmitters. The location state represents the position of camera, is defined as S : where n is index of transmitters:

$$f_n = \begin{bmatrix} x_f \\ y_f \end{bmatrix}, s = \begin{bmatrix} x_s \\ y_s \end{bmatrix}, \qquad (6)$$

Having defined the feature states and location states, the system state, at time t, is then:

$$x_t = \begin{bmatrix} s_t \\ f_{1,t} \\ f_{2,t} \\ \cdots \\ f_{n,t} \end{bmatrix} \qquad (7)$$

Given the above overview of system state, the camera starts moving from an initial position s_0 without prior knowledge of the sensor node, $f_1, f_2, \dots f_n$. As the camera keeps moving it receives relative range data from the sensor transmitter. Using these sensor data the SLAM algorithm tries to estimate the path $s_{0:t}$ of the camera.

The observation model tells the probability of obtaining a camera position at a certain location state. The Bayesian filter can be defined as a probabilistic distribution: $\Pr(d_t \mid s_t)$, where d_t, s_t are the location state and sensor reading, respectively. The straight observation model is given by the following equation:

$$d_s = g(f_s, s) = \sqrt{(x_f - x_s)^2 + (y_f - y_s)^2} + w \quad (8)$$

Where x_f is the coordinate of a frame, x_s is the coordinate of the robot, d is the relative distance from the robot to feature n and w is the Gaussian noise characterizing the errors of the sensors. At each time step, the sensor attached to the camera will receive observation information from all features.

The motion model characterizes the camera location states over time. It helps to predict the next camera location state given the most current one. When implementing the motion model, we have to consider the characteristics of the motion kinematics of the camera. We assumed the target camera trajectory is associated with direction or speed of the movement that is random. Thus we use a 2D Gaussian model to approximate the motion. More specifically, when given the location state s_t at the time step t , to predict the location state s_{t+1} at the time $t+1$, we draw a number of particles randomly from a 2D Gaussian distribution with zero-mean. These particles will form a circle with origin at St and its radius is determined by the standard deviation of the 2D Gaussian distribution.

B. *Particle Filter SLAM Algorithm*

Based on above the system model given above, the data structure of M particles is illustrated in Figure 2:

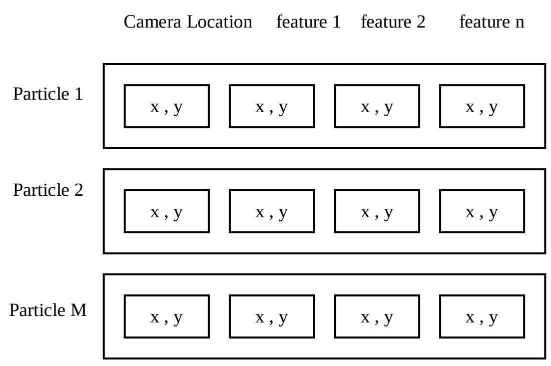


Figure 2 Data structure of particles

Each particle has 2 (n + 1) states: 2 location states and 2n feature states. In a mathematical form, each particle is:

$$\begin{aligned} x_t^m &=< s_t^m, f_{1,t}^m, f_{2,t}^m, \ldots f_{n,t}^m > \\ &=< (x,y)_t^m, (x,y)_{1,t}^m, (x,y)_{2,t}^m, \ldots (x,y)_{n,t}^m > \end{aligned} \quad (9)$$

Where the superscript *m* is the index of the particle, the subscript t indicates the time step, s_t^m is the location *of* the camera and , $f_{n,t}^m$ represents feature *n*. The particle filter algorithm is then operating on a set of particles x_t^m . Each iteration of the algorithm can be divided into the following stages: Initialization, Weighting all the particles, Map Adjustment, Resampling.

Initialization is a most important stage in all SLAM algorithms. In EKF-based SLAM, its task is to initialize the mean and covariance matrix for the state vector, while in this particle filter based SLAM it is to initialize the location state and feature states in each particle. The initialization process can be quite tricky when a single measurement is not enough to constrain a feature location in all dimensions. This problem will bring great ambiguity about the feature states at the beginning of the algorithm. In this research work, an approach is employed to reduce ambiguity by using the first two measurements to obtain a rough idea of where the next location states should be, i.e. in which quadrant the state is. Then a random point is chosen in that quadrant to be the next location state.

After the initialization, the motion model is applied to all particles. Figure 3 is an example showing one particle being applied the motion model.

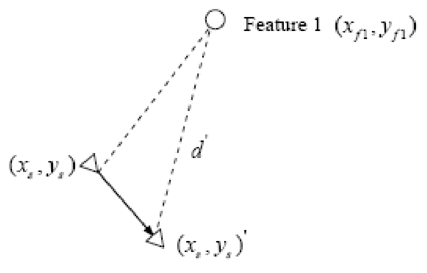


Figure 3: Before applying the motion model, the particle has an estimation of the location state at (x_s, y_s) and estimation of Feature 1 at (x_{f1}, y_{f2}) . After applying the motion model the location state is replaced with $(x_s, y_s)'$ while the estimation to Feature 1 remains unchanged. Only applying the motion model to all particles does not represent the true posterior of the path and features since it does not incorporate the observation. Therefore the weighting process is required which gives individual particle a weight to reflect the observation. Before we describe how to implement the weighting process we need to define some terms: At time step t, before receiving the observation, each particle has its estimation of the location state and feature states. Then we define 'predicted location state' as the location state after being applied the motion model. We also define 'predicted observation' as the distance measurement from the predicted location state to a feature. In a probabilistic mathematical form, the weight of each particle is given by:

$$w^m = \int Pr(d_t \mid f_n, s_t^m) \Pr(f_n \mid s_{0:t-1}^m, d_{0:t-1}) df_n \quad (10)$$

The 'Map Adjustment' is a novel techniques invented in this paper. Its inspiration comes from the 'landmark update' in FastSLAM [15] where the landmark (feature) estimates are updated using EKF. The EKF approach is not suitable in this SLAM problem due to the non-linear and not invertible observation model. The basic idea of Map Adjustment is as follows: For each particle, after applying the motion model and weighting, when the observation is received, each feature's state is then adjusted so that the difference between the predicted observation and real observation is smaller.

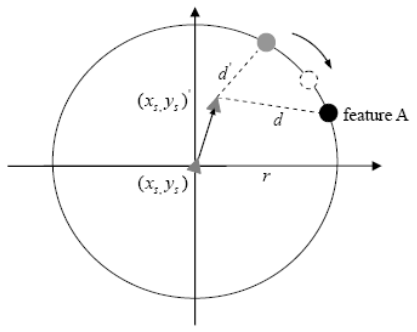


Figure 4 Illustration of the Map Adjustment

In this implementation we use the following equation to calculate the movement:

$$movement = p * \frac{(d - d')}{r} \quad (12)$$

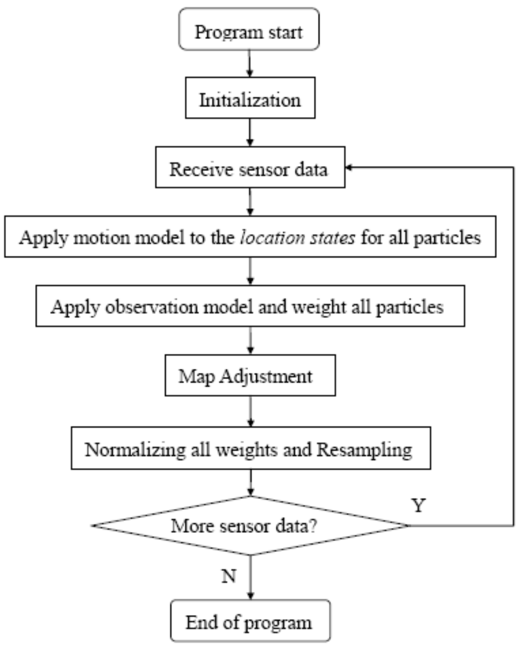
where p is a parameter which must be specified manually based on experiments. By using the Map Adjustment, the accuracy of the estimation to *features* can be greatly improved, or can be maintained but fewer particles are required.

Resampling is the last step in each iteration. This step is very much the same as the one in Particle Filter

Localization. In this process, particles with large weight will be duplicated while those with small weight will be deleted. The sum of all weights of all particles should remain unchanged. Therefore before the resampling a normalization operation is carried out which normalize the weight of all particles so that they sum up to 1.

C. *Algorithm Summary*

The summary of the whole SLAM program, as shown in Figure 5:



Program start
Initialization
Receive sensor data
Apply motion model to the *location states* for all particles
Apply observation model and weight all particles
Map Adjustment
Normalizing all weights and Resampling
More sensor data?
Y
N
End of program

Figure 5 A flowchart of the algorithm

IV. SIMULATION RESULTS

In our research work, we simulate the camera moving trajectory in a virtual environment by using Maya and Matlab. The goal of these experiments is to evaluate the accuracy, robustness and efficiency of this particle filter based SLAM solution, and to investigate if this algorithm has been successfully implemented for 2D camera tracking. In all the simulated datasets, we assumed that there are four fixed features in the sensor network, since four fixed features are easy and simple, Feature A: (10, 10), Feature B: (22, 0), Feature C: (-12, -16), Feature D: (-5 ,15).

A. *Data with Time Steps*

In an experiment a dataset simulated with time steps (120 time steps) is used, to test the stability of this algorithm. A Result is shown in Figure 6 Camera Trajectory estimation with 120 time steps. Figure 7 illustrates the estimation error on path over time. Figure 6 shows that the SLAM algorithm would successfully track the camera trajectory in 2D range with suitable time step successfully. Figure 7 illustrates the errors of the path estimation from time step 0 to 120. At time step 0 since we are assuming that the estimated location and the real location are both at the origin, there are only small errors at the beginning. As the algorithm keeps iterating,

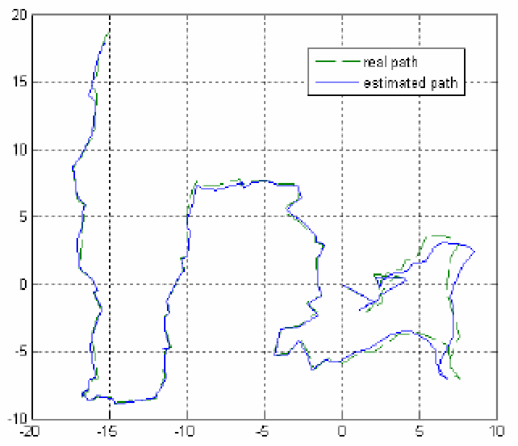


Figure 6 Camera Trajectory

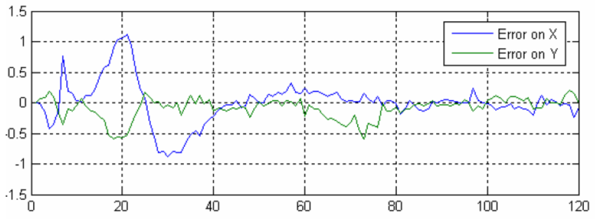


Figure 7 Error on Camera Trajectory over time

We can see there is a significant error at about time step 20. Compare with the real path in Figure 6, we may conclude that this is because the robot changes its direction at that time. Nevertheless, as the camera keeps moving, the errors get smaller and smaller and finally converge. Figure 7 shows that compared with the real feature state, the errors are very little.

B. *Map Adjustment Improvement*

The Map Adjustment technique proposed in this dissertation can help to improve the accuracy, or can maintain the same accuracy with fewer particles are required. Figure 8 illustrates a comparison of two experiments using 200 particles. The left figure is the error of the camera trajectory estimation over time without the Map Adjustment, while the right one is with Map Adjustment. Clearly after applied the Map Adjustment the estimated path converges more quickly to the real path.

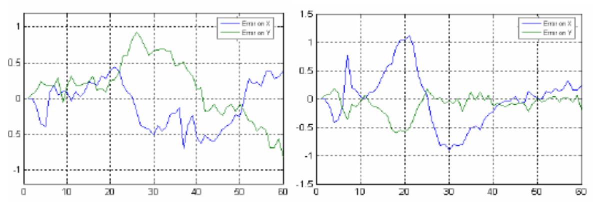
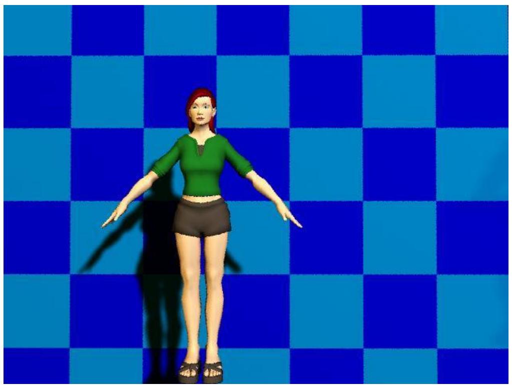


Figure 8 How the Map Adjustment improves the accuracy

C. *Maya Simulation results*

In order to evaluate the algorithm for camera tracking, we use Maya to simulate a virtual world and virtual camera. Based on the time step, we produce the short movie for 2D camera trajectory estimation. The below pictures are the frames from original and estimation camera movement separately. The initial camera position are (0, 0, 4), and the Z axis is instant. The distance of blue Screen from the camera is 20 , and the distance of the character from the camera is 15. The X, Y error of camera in 3D environment does not influence too much. The results show that the algorithm is efficient to 2D camera tracking in the virtual studio.


(a) Original Camera View position A

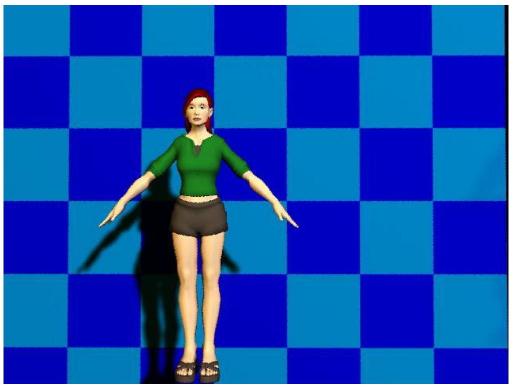

(b) Estimation Camera View Position B

Figure 9 illustrates that one frame in the produced movie (frame 20), from original camera view, the position of camera is A (1.57538, -1.0592, 4) ; from the estimation camera view, the position of camera is B (1.48564, -1.12632, 4). The results show that the error of the camera tracking in virtual studio is not visible for human.

V. CONCLUSION AND FUTURE WORK

Virtual studios have long been used in commercial broadcasting and are starting to evolve into the next level with improved technology in image processing and computer graphics. In this paper, a sensor based SLAM algorithm has been designed and implemented to solve the problem of camera tracking in virtual studio environment. The simulation results show that the algorithm would achieve the research aim successfully. The future work will focus on the extension of the algorithm on orientation estimation and high accuracy in virtual studio. I

REFERENCES

[1]. K. Cornelis, M. Pollefeys, and L. V. Gool. Tracking based structure and motion recovery for augmented video productions. In *Proc. ACM Symposium on Virtual Reality and Software Technology (VRST),* Alberta, Canada, pp. 17-24. 2001.

[2]. M.I. Lourakis and A.A. Argyros, 2005. Efficient, causal camera tracking in unprepared environments. *Computer Vision and Image Understanding* 99, 2 (Aug. '05), pp. 259–290, 2005.

[3]. M. Livingston and A.State: Magnetic Tracker Calibration for Improved Augmented Reality Registration. Presence, vol. 6, pp. 532–546, 1997.

[4].F. Madritsh. Optical Beacon Tracking for Human-Computer Interfaces. PhD thesis, Technical University Graz, 1996. M.Lourakis, Egomotion estimation using quadruples of collinear image points, in: Proc. ECCV'00,vol.2, 2000,pp. 834-848.

[5] R. Want. A. Hopper. V. Falcao. and I. Gibbons. The Active Badge Location System. *ACM Trans. Info.* ***Sys..*** *vol. IO, no. 1,* pp. 91-102. Jan. 1992.

[6] C. Randell and H, Muller Low Cost Indoor Positioning System. *In G. D. Abowd, editor, Ubicomp 2001: Ubiquitou sComputing*, pp. 42–48. Springer-Verlag.

[7] S.Gibbs, C.Arapis, C. Breiteneder, V.Lalioti, S. Mostafaway, and J. Speier, "Virtual studios: An overview, " IEEE Multimedia, vol. 5, no.1, pp.50-57, Jan-Mar. 1998.

[8] A. Wojdala, "Challenges of virtual set technology ," IEEE Multimedia, vol.5, pp. 50-57, Jan-Mar. 1998.

[9] Y. Yamanouchi, H.Mitsumine, T.Fukaya, M. Kawakita, N.Yagi, and S.Inoue, "Real space-based virtual studio – Seamless synthesis of a real set image with a virtual set image, " in *Proc. ACM Symp. Virtual Reality Software and Technology* (VRST), HongKong, Now. 2002, pp. 194-200.

[10] D. Scharstein and R.Szeliski, "A taxonomy and evaluation of dense two-frame stereo correspondence algorithms," *Int. J. Comput. Vis.*, vol. 47, no. 1, pp. 7-42, 2002.

[11] R.Yang, M. Pollefeys, H.Yang, and G.Welch, "A unified approach to real-time, multi-resolution, multi-baseline 2D view synthesis and 3D depth estimation using commodity and graphics hardware," *Int.J.Image Graph*., vol. 4, pp. 1-25, 2004.

[12] A. Ward, A. Jones, and A. Hopper. A New Location Technique for the Active Office. *IEEE Personal Communications*, pp. 42–47, 1997.

[13] R. A. Brooks. A robot that walks: Emergent behavior from a carefully evolved network. *IEEE Journal of Robotics. and Automation*, 2: pp. 253–262, 1989.

[14] S. Thrun, D. Fox, W, Burgard and F. Dellaert. Robust monte carlo localization for mobile robots. *Artificial Intelligence*, vol. 128, no. 1-2, pp. 99–141. 2001.

[15] M. Pupilli and A. Calway. Real-time camera tracking using a particle filter. In *Proc. British Machine Vision Conference*, pp. 519–528,2005.

[16]. A. J. Davison. Real-time simultaneous localisation and mapping with a single camera. In *Proc. IEEE International Conference on Computer Vision*, pp. 1403–1410, 2003.

[17] A.J. Davison and D.W. Murray. Simultaneous Localization and Map-Building Using Active Vision. *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 24, no. 7, pp. 865-880 2002.

[18]. M. H. Degroot and M. J. Schervish, M. *Probability and statistics*. 3rd ed: Addison-Wesley. 2002.