IBIMA Publishing
International Journal of Interactive Worlds
http://www.ibimapublishing.com/journals/IJIW/ijiw.html
Vol. 2010 (2010), Article ID 751615, 14 pages

Markerless Vision-Based Tracking for Interactive Augmented Reality Game

Chutisant Kerdvibulvech

Department of Information and Communication Technology, Rangsit University
52/347 Muang-Ake, Paholyothin Rd. Lak-Hok, Pathum Thani 12000, Thailand

Abstract

In this paper, we present an interactive augmented reality (AR) game for tracking a remote-controlled car controlled by players. We propose it as a new markerless framework for tracking a colored remote-controlled car by integrating a Bayesian classifier into particle filters. This adds the useful abilities of automatic track initiation and recovery from tracking failures in a cluttered background. A Bayesian classifier is utilized to determine the car's color probability before tracking. In addition, by using the online adaptation of color probabilities, this method is able to cope well with luminance changes. We calculate the projection matrix as an online process. The method presented can be used to develop the real-time game of AR to remote-controlled car playing. The application can entertain players interactively by controlling the car to the augmented items. A user study is conducted to evaluate the effectiveness of the aforementioned application.

Keywords: Augmented Reality, Bayesian Classifier, Interactive Game, Particle Filter

Background

Due to the popularity of augmented reality (AR), research about vision-based AR is one of the most popular topics in recent years. Basically, AR is an emerging technology for a live direct or indirect view of a physical real-world environment whose elements are merged with virtual world. AR has been recently applied to many research fields. For instance, Barakonyi and Schmalstieg (2005) built a vision based-system, called Augmented Piano Tutor, to serve as a piano teacher that educates users about basic chords and scales in an AR environment. In addition, Cheng and Robinson (2001) built a system, called Handel (hand-based enhancement for learning) that relies on hand movements to trigger an AR overlay onto the user's hands during piano practice. In their system, a pianist is equipped with a wearable computer system and sits at an acoustic piano with no sheet music. Another example of AR application broadcast by Discovery Science (2009) for entertainment is by showing visual aid information on a real stringed guitar and this becomes an aid to learning to play the guitar. Next, a system which supports the compositing process with AR technology was developed by Berry et al (2003), called The Music Table. Furthermore, a piece of educational software that utilizes collaborative AR to teach users the meaning of Chinese symbols (Kanji) was built by Wagner and Barakonyi (2003). Also, Rohs (2007) presented several marker-based game prototypes that apply ubiquitous passive media, such as

Author contact: dr-chutisant@hotmail.com

product packages, tickets and flyers as background media for handheld augmentation. Another AR research was proposed by Geiger et al (2007) using the virtools dev 3D authoring system and custom extensions. More recently, Mistry et al (2009) from Massachusetts Institute of Technology (MIT) proposed a wearable gestural interface that augments the real world with digital information. It can allow users to use natural hand gestures to interact with that information. Moreover, Lochtefeld et al (2009) presented an AR application for camera projector phones. In their system, the camera projector unit is utilized to augment the hand drawings of a user with an overlay displaying physical interaction of virtual objects with the physical world.

However, these AR systems do not aim to track a remote-controlled car. We have a different goal from most of the previous works. To the best of our knowledge, no previous work has been conducted and applied AR to the remote-controlled car. In this paper, we propose a novel vision-based method for tracking a remote-controlled car. The contribution of this paper is to present a novel framework for tracking a colored object without any marker by combining a Bayesian classifier to particle filters. Our research goal is to accurately determine and track the remote-controlled car position. A challenge for tracking the remote-controlled car is that the car usually moves fast and frequently changes directions. It is therefore necessary to identify the position of the remote-controlled car accurately. Moreover, the background we used is cluttered and non-uniform which makes it more difficult for background segmentation. Another important issue is recovery when the tracking fails. Also, the luminance change makes it more challenging. Our method for tracking the car solves these problems.

We first estimate the projection matrix of the camera by utilizing ARTag (Augmented Reality Tag), presented by Fiala (2005), at every frame. To determine the color probability of car, during the pre-processing we apply a Bayesian classifier that is bootstrapped with a small set of training data and refined through an offline iterative training procedure, as mentioned by Argyros and Lourakis (2004). Online adaptation of car-color probabilities is then used to refine the classifier using additional training images, as described by Argyros and Lourakis (2005). Hence, the classifier is able to deal with luminance changes, even when the amount of light is extremely changed. We then utilize a particle filter, proposed by Isard and Blake (1998), to track the remote-controlled car in 2D space. We propagate sample particles to get the probability of each particle to be on a car based on the color in the image.

After particle filtering, the position of remote-controlled car can be obtained and tracked. Also, the game items are overlaid on the physical world. The game items in this application are coins and mushrooms. Each item has different scores. It can be used to develop instructive interactive AR application for feedback to the players. For playing the game, the players will control the remote-controlled car by looking at the monitor, and at the same time the system will generate the items which are augmented on the screen as well. This application is done by identifying whether the position of car is in accord with the positions of the items we built for the interactive AR game.

Proposed Method

This section will describe the methods implemented. After capturing the images, we calculate the projection matrix in each frame by utilizing ARTag. We then utilize a Bayesian classifier to determine the

car's color probability automatically. Finally, we apply the particle filters to track the position of the car.

Camera Calibration

Our research goal is mainly to robustly detect the position of the car in captured images. Thus, it is necessary to calculate the projection matrix. In the camera calibration process, it is important to compute the projection matrix relative to the world coordinate, as mentioned in the book by Forsyth and Ponce (2002). When both intrinsic and extrinsic camera parameters are known, the camera projection matrix is determined. The important camera properties, namely the intrinsic camera parameters that must be measured, include the principal point, the lens distortion, and the focal length. These intrinsic parameters represent focal length in terms of pixels, as explained in detail by Bayro-Corrochano and Rosenhahn (2002). We compute once the intrinsic parameters during preprocessing. As shown in Equation (1), the matrices R and t in the camera calibration matrix describe the position and orientation of the camera with respect to the world coordinate system. The extrinsic parameters include three parameters for the rotation, and another three for the translation. Using the online process, the ARTag algorithm proposed by Fiala (2005) computes the extrinsic parameters in every frame, and therefore we can compute in real-time the projection matrix, *P*, from both the intrinsic parameters and the extrinsic parameters by using

$$P = A[R,t] = \begin{bmatrix} \alpha_u & -\alpha_u \cot\theta & u_0 \\ 0 & \alpha_v / \sin\theta & v_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} R_{11} & R_{21} & R_{31} & t_x \\ R_{12} & R_{22} & R_{32} & t_y \\ R_{13} & R_{23} & R_{33} & t_z \end{bmatrix} \quad (1)$$

where A is the intrinsic matrix with (u_0, v_0) being the principal point coordinates, α_u and α_v correspond to the focal length and aspect ratio in pixels along the u and v axes of the image, and θ is the skew i.e. the angle between the two image axes. In practice, θ is close to 90° for the real camera. In other words, the intrinsic matrix contains intrinsic parameters which encompass focal length, principal point and image format. In detail, the image format determines the angle of view of a particular lens when used with a particular camera. The larger the image format, the wider the angle of view for a given focal length. Further details about intrinsic matrix can be found in a research study by Wang and Sugisaka (2003).

Adaptive Color Learning

The learning process is composed of two phases. In the first phase, the color probability is learned from a small number of training images during preprocess that is executed only once, as explained by Argyros and Lourakis (2004). In the second phase, we gradually update the probability from the additional training data images automatically and adaptively. The adapting process can be disabled as soon as the achieved training is deemed sufficient, as suggested by Argyros and Lourakis (2005).

During an offline phase, a small set of training input images (20 images) is selected on which a human operator manually segments car-colored regions. The color representation used in this process is YUV 4:2:2. This color representation has also been demonstrated effectively by Jack (2004). This color space contains a luminance component (Y) and two color components (UV). The main advantage for using the YUV space is that the luminance and the color information are independent. Thus, it is easy to separate the chrominance

from the luminance components of the color. Y stands for the luminance component and U and V are the chrominance components. Since car tones differ mainly in color (chrominance) and less in intensity, by employing only chrominance-dependent components of color, one can achieve some degree of robustness to changes in luminance. However, the Y-component of this representation is not employed for two reasons. Firstly, the Y-component corresponds to the luminance of an image pixel. By omitting this component, the developed classifier becomes less sensitive to luminance changes. Secondly, compared to a 3D color representation (YUV), a 2D color representation (UV) is lower in dimensions and therefore, less demanding in terms of memory storage and processing costs. Assuming that image pixels with coordinates *(x,y)* have color values *c = c(x,y)*, training data are used to calculate:

(i) The prior probability *P(r)* of having car color *r* in an image. This is the ratio of the car-colored pixels in the training set to the total number of pixels of whole training images.

(ii) The prior probability *P(c)* of the occurrence of each color in an image. This is computed as the ratio of the number of occurrences of each color *c* to the total number of image pixels in the training set.

(iii) The conditional probability *P(c|r)* of a car being color *c*. This is defined as the ratio of the number of occurrences of a color *c* within the car-colored areas to the number of car-colored image pixels in the training set.

By employing Bayes rule described in the book by Singh et al (2003), the probability *P(r|c)* of a color *c* being a car color can be computed by using

$$P(r \mid c) = \frac{P(c \mid r)P(r)}{P(c)}. \qquad (2)$$

This equation determines the probability of a certain image pixel being car-colored using a lookup table indexed with the pixel's color. The resultant probability map thresholds are then set to be threshold $T_{\max}$ and threshold $T_{\min}$, where all pixels with probability P(r|c) > $T_{\max}$ are considered as being car-colored—these pixels constitute seeds of potential car-colored blobs—and image pixels with probabilities P(r|c) > $T_{\min}$ where $T_{\min} < T_{\max}$ are the neighbors of car-colored image pixels being recursively added to each color blob. The rationale behind this region growing operation is that an image pixel with relatively low probability of being car-colored should be considered as a neighbor of an image pixel with high probability of being car-colored. In other words, an image pixel with relatively high probability should be determined as such in the case that it is a neighbor of an image pixel with low probability. Smaller and larger threshold values cause the false car detection. For example, if we choose the threshold value $T_{\max}$ that is too large, we cannot detect any pixels that constitute the potential car regions. Therefore, the values for $T_{\max}$ and $T_{\min}$ should be determined by test experiments. From existing experimental data, the values for $T_{\max} = 0.5$ and $T_{\min} = 0.15$ give the promising results, in our experiment, so that we use 0.5 and 0.15, for $T_{\max}$ and $T_{\min}$ respectively. These threshold values have also been used in the paper by Argyros and Lourakis (2004). A standard connected component labelling algorithm (i.e. depth-first search) is then responsible for

assigning different labels to the image pixels of different blobs.

The success of the car-color detection depends crucially on whether or not the luminance conditions during the online operation of the detector are similar to those during the acquisition of the training data set. Despite the fact that using the UV color representation model has certain luminance independent characteristics, the car-color detector may produce poor results if the luminance conditions during online operation are considerably different to those used in the training set. Thus, a means for adapting the representation of car-colored image pixels according to the recent history of detected colored pixels is required.

To cope with this issue, car color detection maintains two sets of prior probabilities. The first set consists of P(r), P(c), P(c|r) that have been computed offline from the training set. The second is made up of $P_W(r), P_W(c), P_W(c \mid r)$ corresponding to P(r), P(c), P(c|r) that the system gathers during the W most recent frames respectively. Obviously, the second set better reflects the "recent" appearance of car-colored objects and is therefore better adapted to the current luminance conditions. Car color detection is then performed based on the following weighted moving average formula:

$$P_A(r \mid c) = \gamma P(r \mid c) + (1-\gamma) P_W(r \mid c) \quad (3)$$

where γ is a sensitivity parameter that controls the influence of the training set in the detection process, $P_A(r \mid c)$ represents the adapted probability of a color c being a car color, $P(r \mid c)$ and $P_W(r \mid c)$ are both given by Equation (2) but involve prior probabilities that have been computed from the whole training set [for $P(r \mid c)$] and from the detection results in the preceding W frames [for $P_W(r \mid c)$]. Thus, the car-color probability can be determined adaptively and automatically. Figure 1 illustrates car segmentation based on the car-color probability in the cluttered background using a Bayesian classifier and online adaptation.

Fig. 1. Car segmentation based on the car-color probability.

Automatic Tracking

Spatio-temporal estimation of position over time has been dealt with thoroughly by Kalman filtering, as it can deal with Gaussian densities, as discussed by Gennery (1992) and Harris (1993). However, due to being unimodal, it cannot represent simultaneous alternative hypotheses. In this paper, we apply particle filters, presented by Isard

and Blake (1998), to compute and track the position of the car, with the advantages of performing automatic track initiation and recovering from tracking failures. Automatic track initiation is that, while starting the system, it can be tracked to the position of the car automatically.

The particle filtering (system) uniformly distributes particles all over the input image randomly, and then projects the particles to obtain the probability of each particle to be a car. As new information arrives, these particles are continuously re-allocated to update the position estimate. Each particle demonstrates the probability if it is the car or not. Furthermore, when the overall probability of particles is lower than the threshold we set, the new particles will be uniformly distributed all over the image. In other words, the tracking process will be reset if the overall probability is lower the threshold. Then the particles will move to the pixels of car automatically. For this reason, the system is able to recover the tracking.

As explained by Isard and Blake (1998), given that the process at each time-step is an iteration of factored sampling, the output of an iteration will be a weighted, time-stamped sample-set, denoted by $\{s_t^{(n)}, n=1,\ldots,N\}$ with weights $\pi_t^{(n)}$, representing approximately the probability-density function $p(X_t)$ at time t: where *N* is the size of sample sets, $s_t^{(n)}$ is defined as the position of the n^{th} particle at time *t*, X_t represents the position of remote-controlled car at time *t*, $p(X_t)$ is the probability that a car is at position $X = (x,y)^T$ at time *t*. Details of the probability-density function can be further found in Ribeiro's work (2004).

The number of particles we used, *N*, should be determined from experimental data. If the number of particles used is too large, it will consume time and resources. In this system, the total number of particles is 300 which can produce the promising tracking result in real-time. The iterative process can be divided into three main stages. In the first stage (the selection stage), a sample $s'^{(n)}_t$ is chosen from the sample-set $\{s_{t-1}^{(n)}, \pi_{t-1}^{(n)}, c_{t-1}^{(n)}\}$ with probabilities $\pi_{t-1}^{(j)}$, where $c_{t-1}^{(n)}$ is the cumulative weight. This is done by generating a uniformly distributed random number $rand \in [0, 1]$. We find the smallest j for which $c_{t-1}^{(j)} \geq rand$ using binary search, as described by Cormen et al (2001), and then $s'^{(n)}_t$ can be set as follows: $s'^{(n)}_t = s_{t-1}^{(j)}$. Each element chosen from the new set is now subjected to the predictive step. We propagate each sample from the set s'_{t-1} by a propagation function, $g(s'^{(n)}_t)$, using

$$s_t^{(n)} = g(s'^{(n)}_t) + noise \qquad (4)$$

where noise is given as a Gaussian distribution with its mean = $(0,0)^T$, as explained by Rasmussen and Williams (2006). The value for the variance of the Gaussian distribution of noise should be determined carefully. It is essential to select a suitable value for the variance of the Gaussian distribution of noise because if the variance value is too low, the tracker will easily fail to track, and need to recover too frequently. However, if the variance we set is too high, the accuracy of tracking will decrease. From existing experimental data, the variance of Gaussian distribution = 12 cm^2 can provide the promising results in the experiment.

The accuracy of the particle filter depends also on this propagation function used to

reallocate the particles at each iteration. We have tried different propagation functions, such as constant velocity motion model and acceleration motion model, as suggested by Brasnett et al (2005) and Han et al (2005). Our experimental results have revealed that constant velocity motion model and acceleration motion model do not give an improvement. A possible reason is that the motion of the car is usually quite fast and frequently changing directions while playing the game. For example, the car moves forward quickly and turns left. Then it stops suddenly and moves backward to the right. In this case, the motion of the car is fast and changing directions quickly, so that the constant velocity motion model and acceleration motion model do not provide an improvement. For this reason, we use the noise information by defining $g(x) = x$ in Equation (4).

In the measurement stage, we project these sample particles using the projection matrix results from Equation (1). We then determine the probability whether the particle is on remote-controlled car. In this way, we generate weights from the probability-density function $p(X_t)$ to obtain the sample-set representation $\{(s_t^{(n)}, \pi_t^{(n)})\}$ of the state-density for time t using

$$\pi_t^{(n)} = p(X_t = s_t^{(n)}) = P_A(r \mid c) \qquad (5)$$

where $p(X_t = s_t^{(n)})$ is the probability that a remote-controlled car is at position $s_t^{(n)}$. Following this, we normalize the total weights using the condition $\Sigma_n \pi_t^{(n)} = 1$. Next, we update the cumulative probability, which can be calculated from normalized weights using

$$c_t^{(0)} = 0, \qquad c_t^{(n)} = c_t^{(n-1)} + \pi_t^{(n)}{}_{Total} \quad (n = 1, \ldots, N) \qquad (6)$$

where $\pi_t^{(n)}{}_{Total}$ is the total weight. Once the *N* samples have been constructed, we estimate moments of the tracked position at time-step *t* as using

$$\varepsilon[f(X_t)] = \Sigma_{n=1}^{N} \pi_t^{(n)} s_t^{(n)} \qquad (7)$$

where $\varepsilon[f(X_t)]$ represents the centroid of the remote-controlled car. The car can then be tracked, enabling us to perform automatic track initiation and track recovering even in a cluttered background.

Fig. 2. Coin and mushroom pictures used in the interactive AR game

Interactive AR Game

We develop the AR application to entertain remote-controlled car players, named *Interactive Car Game*, by applying the color tracking method presented in this paper. The reported experiment is based on a sequence that has been acquired. The USB camera with resolution 320x240 has been used. The reported experiment was acquired and processed on an Intel(R) Core (TM) 2 Duo CPU P8600 laptop computer running MS

Windows at 2.4 GHz. The camera captures the remote-controlled car on the floor. In the system, there is one remote-controlled car involved. The type of remote-controlled car we used is wireless. The camera is placed on the table, and captures images which are perpendicular to the floor. Our playing field is in two- dimensions.

The application determines the car position controlled by a player. Then, the system gives real-time feedback to players telling them the score if they can control the remote-controlled car well or not. Our application also contains the voice of music while the players control their car to the correct positions of the items. This voice of music is for entertaining the players while playing. It also contains the voice of music while the players control their car to the correct positions of the items. This voice of music is for entertaining the players. To play the game, the players control the remote-controlled car by looking at the monitor. At the same time, the system will generate the items which are augmented on the screen as well. The items we provided are coins and mushrooms, as depicted in Figure 2. In this way, the players can earn the scores if they match the car to the same positions of the items. In this game, we assign one score (per coin) and two scores (per mushroom), respectively. For example, if the player controls the car to the position of a coin, the system will recognize in real-time and add one score the score to the player. However, the player can have an option to decide to control the car to the position of a mushroom which will earn two scores. After the players can collect their score at ten points, the system will finally evaluate how users perform to control the remote-controlled car. Thus, it can give a feedback to the players (level of efficiency of users). The computation time of our system is around 16 frames per second. We believe this recovering speed is fast enough for real application. This application can be invaluable as a new interactive AR game for remote-controlled car players.

Results and User Studies

In this section, representative results from our experiment are shown. Figure 3 provides a few representative snapshots of the experiment. For visualization purposes, the 2D tracked result of the car is also shown. The particles distributed are shown as well in the left windows.

In the initial stage (frame 10), when the experiment starts, the tracker attempts to find the color which is similar to the car-colored region. The system can perform automatic track initiation because it is using particle filtering. This means that the users do not have to manually define the position of the car before we start the system. The system also shows and augments the objects onto scene which is captured by the camera. The objects we used in this application are coin and mushroom. Later, during frame 15, the player controls the car to collect the mushroom, so that he receives his scores. Next, in frame 28, the player moves his car to collect a coin, thus his scores are increased. Finally, this application shows an overall evaluation score, as a level, indicating the user's skill when the game has been completed which provides greater user friendliness. This is because the players can receive the feedback interactively from the system. For example, this player uses 34 second for collecting 10 points, and therefore the system gives his level as C - Advanced Driver.

After this, we turned off the light source. Therefore, the lighting used to test our system is quite different, if we compare to the lighting while we turned on the light source, as shown in Figure 4. However, the tracked result of car can be still determined without effects of different

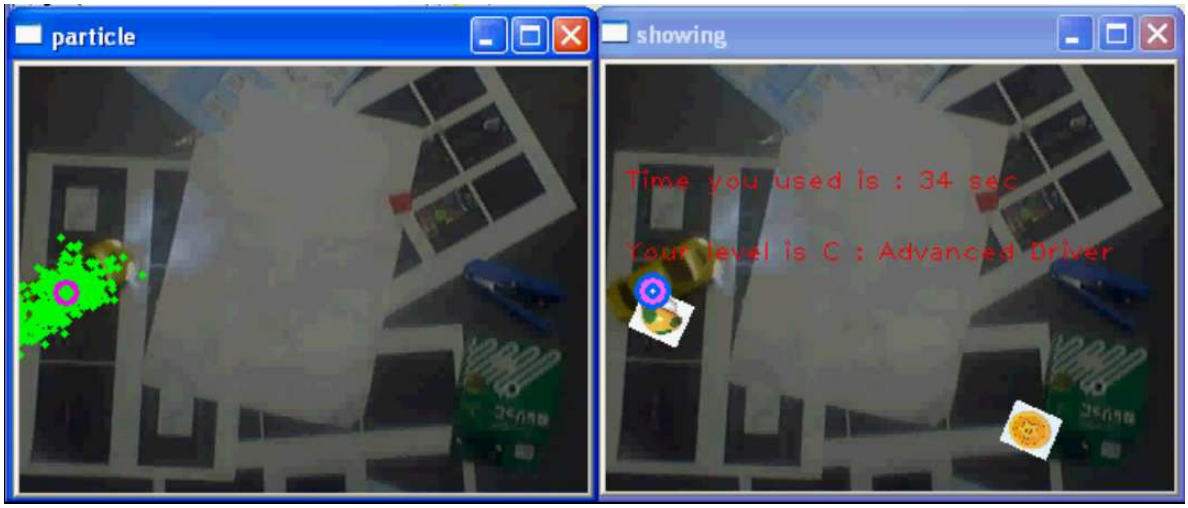
light source. This is because a Bayesian classifier and online adaptation of color probabilities are utilized to deal with this.

Finally, we conducted a user study to evaluate the effectiveness of the aforementioned application. Twenty users were asked to test our system. Each user took approximately 3 minutes to run the system. All users were able to use the system after a short explanation. After the individual tests, the users were asked to give qualitative feedback (Table 1). This included interest in the application, smoothness of the system, user satisfaction, ease of use of the interface, naturalness of the system, and overall impression. General comments on the test were also collected from the users.

Frame 10 (starting the game)

Frame 15 (collecting a mushroom)

Frame 28 (collecting a coin)



This application shows an overall evaluation score, when the game has been completed

Fig. 3. Representative snapshots from the tracking experiment

From this user study, we received mostly positive comments from users. Many users agreed that it is a useful application. Also, they were satisfied with the smoothness and the speed of our system. They reasoned that this was because the system can run in real-time. Moreover, many users indicated that they were impressed by the system, especially by the idea of developing this application. They gave the reason that this was their first time to see how a computer using camera can give a feedback to remote-controlled car players interactively.

Table 1: Qualitative feedback from participants.

Criteria	Subjective feedback (number of respondents)				
	Excellent	**Good**	**Fair**	**Poor**	**Total**
Interest in application	15	5	-	-	20
System smoothness	18	2	-	-	20
User satisfaction	11	9	-	-	20
Ease of use	10	6	4	-	20
System naturalness (comfortable feeling)	12	8	-	-	20
Overall	13	5	2	-	20

However, some people commented about the scope of camera view because the camera is static, and can capture only a limited region. Some users indicated that, although they preferred the idea of this application, it was slightly difficult to use because the scope of camera view was limited because we use the normal camera. A possible solution could be to use a special camera (e.g. high speed camera) and allow it to move dynamically. However, in our case, we would like to focus on using the normal camera. Thus, the system seemed to be slightly not easy to use in some practical games. This was the reason why they gave lower scores to ease of use, as presented in Table 1.

Frame 8 (starting the game)

The game has been completed

Fig. 4. Representative results while turning the light off

Conclusions

We have developed an interactive system that tracks the position of the remote-controlled car accurately in this paper. A novel framework for colored remote-controlled car tracking has been proposed based on a Bayesian classifier and particle filters. ARTag has also been utilized to calculate the projection matrix. This implementation can be used to develop AR application using computer vision technology. It can interactively entertain a remote-controlled car player in real-time by employing computer vision aid as a new AR application. The aforementioned application would assist players of the remote-controlled car by giving them a feedback automatically. To the best of our knowledge, no previous work has been applied AR to the remote-controlled car as the proposed AR application.

Even though we believe that we can successfully produce an accurate tracking system output, the current system has the limitation about the ease of use because the scope of camera view is limited in our current system. In other words, the system only works with a top-down camera view. This sometimes makes it not so easy to use for some users, especially children in real life. As future work, we intend to make technical improvements to further refine this problem.

References

Argyros, A. A. and Lourakis, M. I. A. (2004), 'Real time Tracking of Multiple Skin-Colored Objects with a Possibly Moving Camera,' Proceedings of the European Conference on Computer Vision (ECCV), 2004, Prague, Czech Republic, Springer-Verlag (3)368-379.

Argyros, A. A. and Lourakis, M. I. A. (2005), 'Tracking Multiple Colored Blobs with a Moving Camera,' Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR), 2005, San Diego, CA, USA, 2(2), 1178.

Barakonyi, I. and Schmalstieg, D. (2005), 'Augmented Reality Agents in the Development Pipeline of Computer Entertainment,' Proceedings of the International Conference on

Entertainment Computer (ICEC), 2005, Sanda, Japan, 345-356.

Bayro-Corrochano, E. and Rosenhahn, B. (2002), 'A geometric approach for the analysis and computation of the intrinsic camera parameters,' *Pattern Recognition*, 35, 169-186.

Berry, R., Makino, M., Hikawa, N. and Suzuki, M. (2003), 'The augmented composer project: the music table,' Proceedings of the IEEE and ACM International Symposium on Mixed and Augmented Reality (ISMAR), 2003, Tokyo, Japan, 338-339.

Brasnett, P., Mihaylova, L., Bull D. and Canagarajah, N. (2005), 'Sequential Monte Carlo Tracking by Fusing Multiple Cues in Video Sequences,' *Image and Vision Computing*, 25-8, 1217-1227.

Cheng, L. and Robinson, J. (2001) 'Personal Contextual Awareness through Visual Focus,' *IEEE Intelligent Systems*, 3(16), 16-20.

Cormen, T. H., Leiserson, C. E., Rivest, R. L. and Stein, C. (2001), Introduction to Algorithms, Second Edition, Massachusetts Institute of Technology (MIT) press, ISBN 0262032937, 1184.

Discovery Science (2009), in the programme series 'Popular Science's Future of Play,' from *Discovery Channel*, [Retrieved October 18th, 2009], http://science.discovery.com/videos/popscis-future-of-augmented-reality.html and http://www.youtube.com/user/ScienceInDiscovery

Fiala, M. (2005), 'Artag, a Fiducial Marker System Using Digital Techniques,' Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR), 2005, San Diego, CA, USA, (1) 590-596.

Forsyth, D. A. and Ponce. J. (2002), Computer Vision: A Modern Approach, Prentice Hall, ISBN 0-130-85198-1.

Geiger, C., Stoecklein, J., Klompmaker, F. and Fritze, R. (2007), 'Development of an Augmented Reality Game by Extending a 3D Authoring System,' Proceedings of the ACM SIGCHI international conference on Advances in computer entertainment technology, 2007, Salzburg, Austria, 230-231.

Gennery, D. B. (1992), 'Visual tracking of known three-dimensional objects,' *International Journal of Computer Vision (IJCV)*, ISSN 0920-5691, 7(3), 243-270.

Han, B., Yang, C. Duraiswami, R. and Davis, L. 'Bayesian Filtering and Integral Image for Visual Tracking,' Proceedings of the International Workshop on Image Analysis for Multimedia Interactive Services (WIAMIS), 2005, Montreux, Switzerland.

Harris, C. (1993), Tracking with rigid models, active vision, Massachusetts Institute of Technology (MIT) press, ISBN 0-262-02351-2, 59-73.

Mistry, P., Maes, P. and Chang L. 'WUW - Wear Ur World - A Wearable Gestural Interface,' Proceedings of the ACM Conference on Human Factors in Computing Systems (CHI), 2009, Boston, MA, USA.

Isard, M. and Blake, A. (1998), 'Condensation - conditional density propagation for visual tracking,' *International Journal of Computer Vision (IJCV)*, 29(1), 5-28.

Jack, K. (2004), Video Demystified. Elsevier Science, United Kingdom.

Lochtefeld, M., Schoning, J., Rohs, M.and Kruger A. 'LittleProjectedPlanet: An Augmented Reality Game for Camera

Projector Phones,' Proceedings of the Workshop on Mobile Interaction with the Real World (MIRW) at MobileHCI, 2009, Bonn, Germany.

Rasmussen, C. E. and Williams, C. K. I. (2006), Gaussian Processes for Machine Learning, Massachusetts Institute of Technology (MIT) press, ISBN 0-262-18253-X.

Ribeiro, M. I. (2004), Gaussian probability density functions: Properties and error characterization, Technical Report, Institute for Systems and Robotics, Instituto Superior Tcnico, Portugal.

Rohs, M. (2007), 'Marker-Based Embodied Interaction for Handheld Augmented Reality Games,' *Journal of Virtual Reality and Broadcasting (JVRB),* 4(5), ISSN 1860-2037.

Singh, R., Warmuth, M. K., Raj, B. and Lamere, P. (2003), 'Classification with free energy at raised temperatures,' Proceedings of the European Conference on Speech Communication and Technology (EUROSPEECH), 2003, Geneva, Switzerland, 1773-1776.

Wagner, D. and Barakonyi, I. (2003), 'Augmented Reality Kanji Learning,' Proceedings of the IEEE and ACM International Symposium on Mixed and Augmented Reality (ISMAR), 2003, Tokyo, Japan, ISBN 0-7695-2006-5, 335.

Wang, J. and Sugisaka, M. (2003), 'Camera calibration for a mobile robot prototype,' *Journal Artificial Life and Robotics*, ISSN 1433-5298 (Print) 1614-7456 (Online), 7(3), 91-94.