Visual object tracking in 3D with color based particle filter

Pablo Barrera, José M. Cañas, Vicente Matellán

***Abstract*— This paper addresses the problem of determining the current 3D location of a moving object and robustly tracking it from a sequence of camera images. The approach presented here uses a particle filter and does not perform any explicit triangulation. Only the color of the object to be tracked is required, but not any precise motion model. The observation model we have developed avoids the color filtering of the entire image. That and the MonteCarlo techniques inside the particle filter provide real time performance. Experiments with two real cameras are presented and lessons learned are commented. The approach scales easily to more than two cameras and new sensor cues.**

***Keywords*— Monte Carlo sampling, multiple view, particle filters, visual tracking.**

I. Introduction

OBJECT tracking is a useful capability for autonomous systems like ambient intelligence or mobile robotics, and even for computer-human interaction. Cameras are cheap and ubiquitous sensors. Images may provide much information about the environment, but usually it takes lot of computing power to extract relevant data from them. A basic information they may provide is the 3D location of an object or person who is moving around the camera environment.

Many commercial applications may take benefit from a robust object tracking. For instance, Gorodnichy et al [1] employ a tracking system which allow a person to use her nose as a mouse in front of a personal computer endowed with two off-the-self cameras. In security applications, unsupervised cameras may autonomously track moving persons and trigger an alarm if the person approached to any protected location.

The object tracking techniques may be also used in robotics to cope with the self-localization problem. If the mobile robot tracked the relative 3D positions of some surrounding objects, and their absolute locations are known, then it could infer its own position in such absolute frame of reference. Such objects may not be dynamic, but the robot's motion causes a relative movement which demands a tracking. Davison work [2] is a good example for this. Actually, object tracking and localization share much mathematical background with dynamic state estimation like Kalman filters, probabilistic grid based methods [3] and MonteCarlo sampling methods [4], [5]. There are also approaches to the visual 3D tracking based on genetic algorithms [6][7], similar in spirit to the sequential MonteCarlo techniques but without the probabilistic foundations.

Manuscript received April 26, 2005. Support for this work has been provided by project DPI2004-07993-C03-01 from Spanish Education and Science Ministry

P. B. Author is supported by a research grant from Universidad Rey Juan Carlos

J.C. and V. M. Authors are with the Universidad Rey Juan Carlos, Móstoles, España. E-mail: jmplaza@gsyc.escet.urjc.es .

The approach described here uses a particle filter based on the CONDENSATION algorithm [8], but in a different scenario from the contour tracking inside an image. A similar approach [9] has been recently followed to track objects inside images based on movement, color and speech cues. While our algorithm shares the 2D observations (the images), it focuses on a 3D tracking and uses only color. It requires calibrated cameras and uses projective geometry to forward project particles into all the images. The color of such projection and its neighbors provides feedback about the closeness of the particles to the real 3D location of the coloured object. A gaussian random noise is used as the motion model of the particles. Such model allows the particle population to follow any object movement.

The rest of the paper is organized as follows. Second section explains our particle filter, detailing the observation and motion models used. Third section introduces the experimental setup and some tests of the system. Finally some conclusions and future lines are sketched out.

II. Color based particle filter for 3D tracking

Our approach uses the CONDENSATION algorithm [8] to estimate location of a coloured object. This is an iterative algorithm which includes three steps on each iteration: prediction, update and resampling. CONDENSATION is a Bayesian recursive estimator which uses Sequential MonteCarlo Importance Sampling.

In short, it estimates the current multidimensional state $X(t)$, using a collection of sequential observations $[obs(t), obs(t-1), obs(t-2)..., obs(t_0)]$. The observations are related to the state through a probabilistic observation model $p(X(t)|obs(t))$. The state itself may be dynamic, and such dynamism is captured in a motion model $p(X(t)|X(t-1))$. The sequential nature of the algorithm provides iterative equations, and its sampling nature makes it to manage a set of N particles to represent the $p(X(t)|obs(t), obs(t-1)...)$. A more rigorous and broad description of probabilistic estimators can be found in [5] and [10].

Each particle $s_i(t)$ represents an state estimate and has a weight $w_i(t)$ associated, regarding the importance sampling. Global estimates can be made from the whole particle set, like choosing that of the higher weight (Maximum a Posteriori) or a weighted mean (Minimum Mean Square Error).

The *prediction step* in each iteration of CONDENSATION samples the motion model for every particle, obtaining a new $s_i(t)$, and so building a new particle set. In the *update step*, the

weights of all particles are computed following the observation model: $w_i(t) = p(s_i(t)|obs(t))$. Those particles which are likely given the current observation increase their weights. In the *resampling step*, a new set of particles is built sampling from the weighted distribution of current particles. The higher the weight, the more likely that particle appear in next set. Full details are provided at the original paper [8].

In our approach the state to be estimated is the 3D location of the object $X = [x, y, z]$, so the particles have the shape of $s_i(t) = [x_i(t), y_i(t), z_i(t)]$, they are 3D positions. The observations are just the color images on M cameras and the motion model is just a simple one that randomly move the particles through the three dimensions following a gaussian distribution for each step.

A. Movement model

Similar to [11], our approach uses a weak motion modelling, in order to accomodate to any real movement of the object. This provides robustness to the tracking algorithm as it avoids the need of a precise movement modelling to perform properly.

The motion model is a gaussian distributed one, with the same typical deviation σ_m for x, y and z axis. It follows the equations (1), (2) and (3). There is no privileged motion direction, as the object may equally move in any of them. The size of σ_m has influence on the particle speed while walking inside the state space.

$$x_i(t) = x_i(t-1) + N(0, \sigma_m) \quad (1)$$
$$y_i(t) = y_i(t-1) + N(0, \sigma_m) \quad (2)$$
$$z_i(t) = z_i(t-1) + N(0, \sigma_m) \quad (3)$$

B. Observation model

The update step gives the new weights of the particles according to the last sensor observation. Our observation model is color based and works with any number M of cameras. It takes each camera separatedly, treates them as if they were independent observations and so multiplies all the partial conditioned probabilities. For two cameras it takes the form of (5).

$$w_i(t) = p(s_i(t)|img_1(t), img_2(t), \ldots) \quad (4)$$
$$w_i(t) = p_1(s_i(t)|img_1(t)) * p_2(s_i(t)|img_2(t)) \quad (5)$$

Each individual conditioned probability like $p_1(s_i(t)|img_1(t))$ is computed as follows. First, we project the particles into the corresponding image plane using a pinhole camera model. We assume cameras have no distortion.

- If such projection falls outside the image limits, then $p_1(s_i(t)|img_1(t)) = 1/25$
- If the projection falls inside the image limits, its vicinity is explored to count the number m of pixels with a color similar to the target color. The vicinity is a 5x5 window around projected pixel. This can be seen in Fig. 1. The equation (7) assigns a probability proportional to m. To avoid probability locks with zeroes and to tolerate occlusions, m is set to 1 when no pixel matches the target color description.

$$outside : p_1(s_i(t)|img_1(t)) = 1/25 \quad (6)$$
$$inside : p_1(s_i(t)|img_1(t)) = m/25 \quad (7)$$

The color is described in HSI space which is more robust to changes in illumination than RGB. A target color is defined with two pairs H_{min}, H_{max} and S_{min}, S_{max}. Pixels with very low or very high intensity are silenty discarded and do not match any color description.

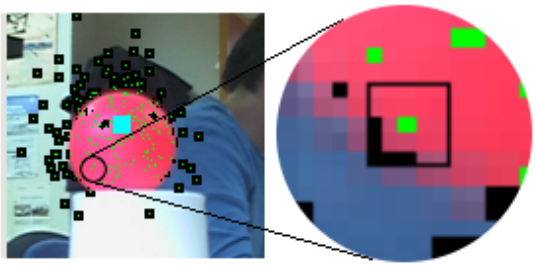

Fig. 1. 5x5 vicinity window for observation model computation

The observation model in (5) clearly rewards those 3D locations which are compatible with several cameras simultaneously. In the case of two, the 3D locations compatible with one camera but which project badly in the other score poorly, because $p_1(s_i(t)|img_1(t))$ or $p_2(s_i(t)|img_2(t))$ is set to a minimum, and that keeps the $w_i(t)$ at small values. This combined reward will lead the particles to the right 3D positions.

Another advantage of this observation model is that it avoids the need to color filtering of the entire image. Depending on the number of particles this can be convenient and reduce the number computations. In our experimental setup, for instance, filtering the whole image requires 320x240 pixel evaluations and the model requires Nx25 pixel evaluations. So for $N < 3072$ it is worthwhile.

In addition the observation model doesn't require any segmentation in the images neither the search for salient points.

C. Considerations

Our approach requires calibrated cameras, but no back-projection or triangulation is performed. Only the forward projections, from 3D particles into image planes, are used. Actually, there is no matching between the stereo images, no correlation involved, and no explicit triangulation are carried out. The observation model rewards those 3D locations with are color compatible in all images. This may include more space areas than the true one, and may lead to the particle cloud to be splitted into such areas. This reflects the fact that particle filter can represent multiple simultaneous hypothesis about the state. New observations will eventually break the ambiguity and the population will converge to the real object position.

The developed algorithm is a true multi-image algorithm [12]: there is no privileged camera, all images are treated equal and it may be used with an arbitrary number of cameras.

III. Experiments

The algorithm has been tested in our lab with two real cameras to track a pink ball. The setup is shown in Fig. 2, where camera 1 is located at $(0.5, 0.0, 0.195)$ (m) and camera 2 at $(0.07, 0.485, 0.085)$ (m) of that coordinate system. The cameras are two webcams, which have been calibrated using OpenCV library. Their external parameters like absolute position and orientation have been manually adjusted using a tape measure and projecting an absolute 3D grid into the images. They provide 320x240 color images through the video4linux API. Right camera was rotated 90^o so it delivers 240x320 images.

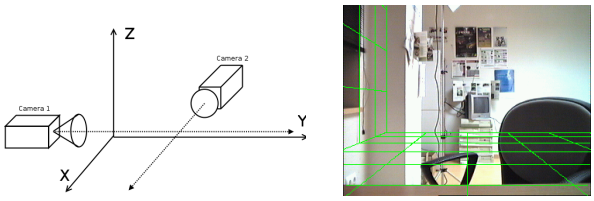

Fig. 2. Experimental setup (left) and projected grid (right)

The particle filter has been tuned to 200 particles, and a typical deviation $\sigma_m = 0.03$ (m). The vicinity window for observation model was set up to 5x5 pixels, as described previously. A typical filter iteration including all the prediction, update and resampling steps takes around 5 ms (on a Pentium IV, 2'7 GHz with HyperThreading) which is enough to real time performance.

A. Typical execution

The Fig. 3 shows a regular run of the particle filter displaying their projection in both images at three different times (iteration 2, 50 and 60). The Fig. 4 shows the projection of the same particle cloud in the XY plane.

The particles are initially located at position $(0.4, 0.1, 0.2)$ (m), just in front of the camera 1 (this initalization will be justified later on). In two iterations they spread following the gaussian motion model. As can be seen in the upper pair of Fig. 3, particles project around the pink ball for the left camera (camera 1), but are out of scope of the right camera (camera 2).

After 50 iterations, the particle cloud has moved itself away from the camera 1, along its optical axis and always keeping their projections around the pink ball in such camera. In Fig. 4, the typical deviation in Y (optical axis of camera 1) is greater than in X. Also, the middle pair of Fig. 3 shows some of the particles entering inside the scope of right camera while keeping around the ball in left image.

Finally, at 60th iteration the particles converge around real 3D location of the ball, and their projections into both cameras are coherent with the ball. If the ball doesn't move, the population remains stable around its real position. Once the population has converged, smooth movements of the ball are successfully tracked in any direction. It can be noticed that convergence of the whole population speeds up as soon as some particles enter into the ball projection.

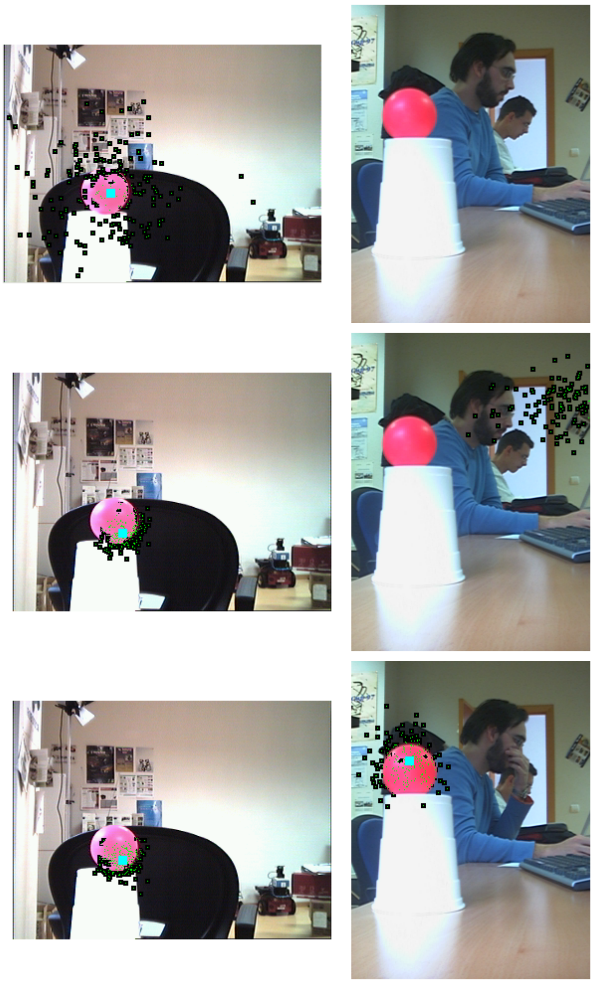

Fig. 3. Particle projections in left (camera 1) and right cameras (camera 2) at iterations 2, 50 and 60

The position error is defined as the distance between the position estimate from the particle filter and the ground truth position of the tracked object. In all the experiments such error lied under 3 cm after the particle set has converged. Low position error means that the particle cloud has settled around the right location.

B. Systematic drift

We have observed a systematic pernicious trend of the particles to move far away from the cameras along their optical axis. This can be noticed in the Fig. 4. Experiments were also carried out with random initalization, and starting the particles at a point further than the ball to a given camera. All such runs resulted in no convergence at all: the systematic drift evolved the particles cloud consistently with one image, but always moving away from the other.

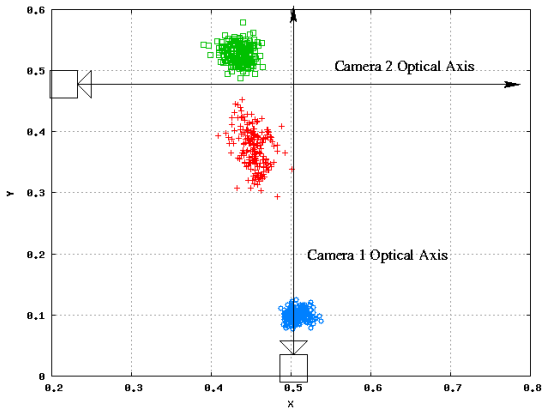

Fig. 4. Three eyebird snapshots of the particles at iterations 2, 50 and 60

Given the conic shape of the projective observation model, after the prediction step there is equal chance to fall closer or further to the camera than the current location, but falling closer makes less likely to project inside the image, makes harder to achive a good observation likelihood, and so, smaller the chance of surviving after the resampling. This can be seen in Fig. 5.

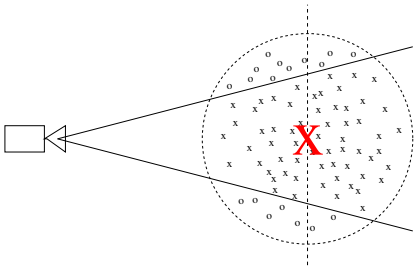

Fig. 5. Particle and its likely positions at $t+1$

C. Particle filter dynamics

In a different set of experiments we have studied the particle filter dynamics, in particular we were interested in its convergence speed. It was studied measuring the evolution of the position error of the filter when the tracked objet suddenly moved to a different location. Instead of the 2D observations described we used the pink ball 3D position estimate as observations for the particle filter. A 3D gaussian observation model was developed for such observations. In real experiments such estimates were built triangulating the centers of masses of the pink pixels on each camera image.

In Fig. 6 the evolution of the position error is displayed. The vertical axis means the position error in (m), and the horizontal axis represents time, in iterations of the filter. The object is stable at the initial position until 30th iteration, and then moves to a new location 2 meters away. To avoid the effect of noise in the observations while studying the filter dynamics, in this experiment they were simulated and set to the ground truth 3D position of the pink ball.

As can be seen in Fig. 6, it takes 5 iterations to the filter to converge at the initial position of the object. At the 30th iteration the error reaches 2 meters, just the distance that the object has moved. Then the filter needs around 15 iterations more to converge at the new object location.

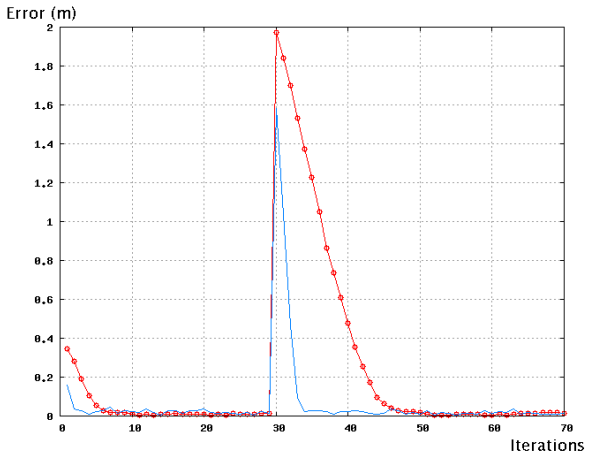

Fig. 6. Time evolution of the position error

The convergence speed of the particle filter has been studied for different σ_m values, in order to determine the right value for such parameter. In Fig. 7 the evolution of the position error is displayed. The horizontal axis represents the time, from the initial iteration at the left and increasing number of iterations to the right. The vertical axis represents different values of σ_m, from 0.001 (m) to 0.3 (m) at regular increments upwards. The pixel color represents the position error, the darker the higher. Colors close to black mean the filter estimation of position is far away from real one, colors similar to white mean they are close. The tracked object moved suddenly at 30th iteration. This way, each row shows an evolution like that in Fig. 6, but encoded in color.

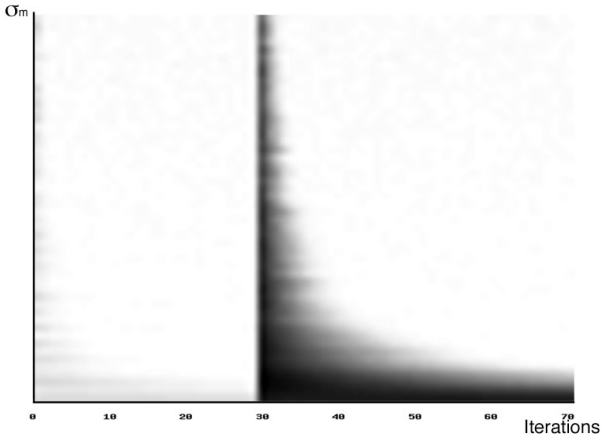

Fig. 7. Evolution of position error at different σ_m values

The experiments in Fig. 7 show that high values of σ_m speed up the convergence. The time needed to find the new object location is the dark gap after the 30th iteration, inside each row. As σ_m increases, such time asymptotically decreases. For very low values of σ_m, the particle filter is not able to find

the new object position before 70th iteration, as can be seen in the left side of the Fig. 7.

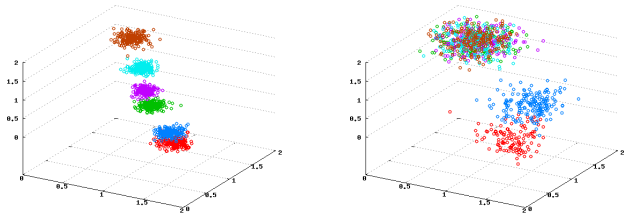

Fig. 8. Spatial evolution of the particle set

But such improvement in convergence speed does not come for free. We have observed that for high values of σ_m the typical deviation of the particle cloud increases. In Fig. 8 the evolution for two particle set is displayed, with snapshots of the particle clouds at six different instants. The same color means the same iteration in both filters. With small σ_m values (left), the population slowly approaches to the location of the object, keeping itself compact. With high σ_m values (right) the cloud reaches sooner the new position of the tracked object, but it is spread over a wide area.

IV. Conclusions and Future Lines

The work presented here summarizes the preliminary results on particle filter for object 3D tracking based on color information. The algorithm doesn't need any explicit triangulation or stereo matching at all, and it scales to an arbitrary number of cameras. The observation model used avoids the color filtering of the whole images and looks at the vicinity of the particle projections to estimate the particle's likelihood.

The results are promising as convergence has been validated in real experiments and the algorithm implementation exhibits real time performance. The real location of the object is an stable point for the particle cloud, and the particles successfully track smooth movements of the object. An interesting systematic drift in the particle behavior has been discovered and explained.

The experiments carried out are just a proof of concept. More experiments are necessary in order to validate the algorithm. Further improvements of the algorithm are coming. First, the use of more than 2 cameras simultaneously, in order to expand the volume inside which objects are succesfully tracked. Second, we are also exploring some proposal distributions inside the filter which hopefully would increase convergence speed of the cloud and its recovery capacity in case of losing the object.

References

[1] D. Gorodnichy, S. Malik and G. Roth, *Affordable 3D face tracking using projective vision*, Proc. of Int. Conf. on Vision Interface, pp. 383-390, Calgary (Canada) May 2002.

[2] A.J. Davison, Y. González-Cid and N. Kita, *Real-time 3D SLAM with wide-angle vision*, 5th IFAC/EURON Symposium on Intelligent Autonomous Vehicles, July 2004.

[3] D. Margaritis and S. Thrun, *Learning to locate an object in 3D space from a sequence of images*. Proc. of Int. Conf. on Machine Learning, pp. 332-340, 1998.

[4] D. Fox, W. Burgard, F. Dellaert and S. Thrun, *Monte Carlo localization: efficient position estimation for mobile robots*, In Proc. of 16th. AAAI Nat. Conf. on Artificial Intelligenge, pp. 343-349, Orlando (USA), July 1999

[5] S. Arulampalam, S. Maskell, N. Gordon and T. Clapp, *A tutorial on particle filters for on-line non-linear/non-gaussian bayesian tracking*, IEEE Transactions on Signal Processing, vol. 50, no. 2, pp. 174-188, 2002.

[6] A.M. Boumaza and J. Louchet, *Mobile robot sensor fusion using flies* In Günter Raidl et.al. editors, Applications of Evolutionary Computing, EvoWorkshops 2003, Lecture Notes in Computer Science, vol. 2611, pp. 357-367, Springer, 2003.

[7] Jean Louchet, *Using an individual evolution strategy for stereovision*, Genetic Programming and Evolvable Machines, vol. 2, pp. 101-109, 2001

[8] M. Isard and A. Blake, *CONDENSATION- conditional density propagation for visual tracking*, Int. Journal of Computer Vision, vol. 20, no. 1, pp. 5-28, 1998.

[9] P. Pérez, J. Vermaak and A. Blake, *Data fusion for visual tracking with particles*, Proceedings of IEEE, vol. 92, no. 3, pp. 495-513, March 2004.

[10] D. Mackay, *Introduction to Monte Carlo methods*, In M. Jordan editor, Learning in graphical models, pp. 175-204, MIT Press, 1999

[11] A. J. Davison, *Real-time simultaneous localisation and mapping with a single camera*, IEEE Int. Conf. on Computer Vision, ICCV-2003, pp. 1403-1410, Nice (France), October 2003.

[12] R.T. Collins, *Multi-image focus of attention for rapid site model construction*, IEEE Int. Conf. on Computer Vision and Pattern Recognition, 1997, pp. 575-581, San Juan, Puerto Rico, June 1997.