Towards Probabilistic Shape Vision in RoboCup: A Practical Approach

Sven Olufs, Florian Adolf, Ronny Hartanto, and Paul Plöger

Department of Computer Science, Bonn-Rhein-Sieg University of Applied Sciences,
D-53757 St. Augustin, Germany
sven@olufs.com, paul.ploeger@fh-brs.de

Abstract. This paper presents a robust object tracking method using a sparse shape-based object model. Our approach consists of three ingredients namely shapes, a motion model and a sparse (non-binary) subsampling of colours in background and foreground parts based on the shape assumption. The tracking itself is inspired by the idea of having a short-term and a long-term memory. A lost object is "missed" by the long-term memory when it is no longer recognized by the short-term memory. Moreover, the long-term memory allows to re-detect vanished objects and using their new position as a new initial position for object tracking. The short-term memory is implemented with a new Monte Carlo variant which provides a heuristic to cope with the loss-of-diversity problem. It enables simultaneous tracking of multiple (visually) identical objects. The long-term memory is implemented with a Bayesian Multiple Hypothesis filter. We demonstrate the robustness of our approach with respect to object occlusions and non-Gaussian/non-linear movements of the tracked object. We also show that tracking can be significantly improved by using compensating ego-motion. Our approach is very scalable since one can tune the parameters for a trade-off between precision and computational time.

1 Introduction

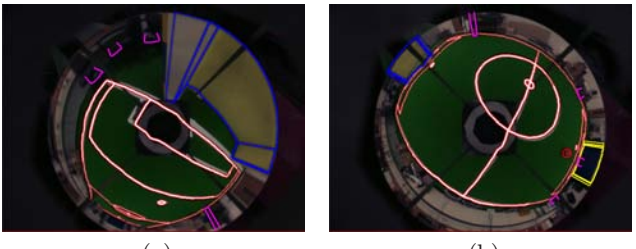
The ability of knowing "where is what?" seems to be easy for humans while it is a cognitive challenge for a machine. The knowledge about *object tracks* (e.g., position, movement) is one of the key capabilities for a high level behaviour. In a typical RoboCup MSL scenario this means that we want to detect and track static and dynamic objects, e.g. a ball, goals, corner-posts, other robots or humans. Additionally we must cope with the object dynamics and a rapidly changing environment.

We can distinguish two major components in a typical visual tracker: (1) *Target Representation and Localisation* and (2) *Filtering and Data Association*. (1) is mostly a bottom-up process which has to cope with changes in the appearance of the target, like for example Mean-Shift [5,3], Particle filtering [10,11] or iterative error-minimising [8] approaches. Those kinds of approaches typically keep the position of the object of interest in an image sequence. (2) is mostly

G. Lakemeyer et al. (Eds.): RoboCup 2006, LNAI 4434, pp. 171–182, 2007.

a top-down process dealing with the *dynamics* of the tracked object, e.g. like Probabilistic Data Association filters (PDAF) [1], Sample-Based Joint Probabilistic Data Association Filters (SJPDAF) [15] or Multiple Hypothesis Tracking (MHT) [14,6] approaches. Those kind of filters are typically dealing with an abstract (anonymous) observation and solve the *tracking and data association problem* [1].

Solely *bottom-up* approaches [5,3,10,11,8] provide a limited robustness with respect to high dynamics (e.g. a bouncing ball), occlusions, visually hidden or disappearing gone objects. Usually, it assumes that the initial position of the object of interest is known a-priori and no visually identical objects appear in the scene. In fact, this assumption does not hold true for a typical RoboCup scenario where we have two goals and multiple players on the field. Recovering lost objects and tracking of multiple objects are typically the domains of *top-down* approaches using (primitive) classifiers. The idea of using a binary homogeneity criterion (e.g. binary colour segmentation) as the primitive classifier is still popular within the RoboCup community [9], though it is bound to the RoboCup soccer domain and not directly applicable to real world applications.



(a) (b)

Fig. 1. Contours of the detected and tracked objects and the projected believe pose of the robot. The tracker has no knowledge about the initial object poses.

Our new probabilistic approach is a hybrid solution of both concepts using a Bayesian *dynamic state space* formulation, see [7] for theoretical issues. Note that we are not interested in solving the *data association* problem. Basically our approach is inspired by the idea of a *short-term* and a *long-term memory*. The objective of the short term-memory is to track objects through image sequences. An observation is immediately lost if the object is no longer detected in the image. The objective of the long-term memory is to maintain tracks of the recognised object coming from the short-term memory. Its role is to re-detect an object if it is no longer detected by the short-term memory. The component gives a feedback to the short term memory in case of a "missing" object. Technically the long-term memory is implemented using a Bayesian *Multiple Hypothesis Tracker*. The short-term memory is implemented by using a new extended method of Bayesian Monte Carlo filters, namely the *extended Particle Filter* method. This extension allows obtaining multiple independent states estimation. We show that our tracking approach is more robust for fast moving

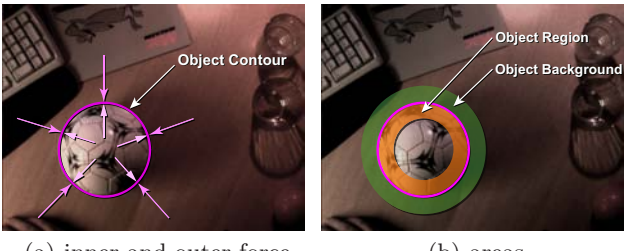
objects than exclusively used (standard) approaches. We also show that ego motion compensation can improve the performance and robustness of the tracker considerably. In order to be applicable to the RoboCup domain we assume all motions of robots and objects are non-Gaussian/non-linear. Furthermore we assume that any initial position of the object of interest is a-priori unknown and that visibility is not known. Finally we assume that multiple visually identical objects can appear in the image.

The current implementation of our approach was tested extensively for coloured objects in various RoboCup soccer competitions as well as on real-world soccer playgrounds. Furthermore our method was successfully applied to other tasks like mobile vehicle surveillance and multi-target tracking of real-world objects in cluttered scenes.

The paper is organised as follows. Section 2 introduces the probabilistic *object model* used by the short-term memory. In section 3 a practical approach is presented. Experimental results are shown in Section 4. Finally we discuss the results and the approach in Section 5.

2 Object Model

The **pose** of a tracked object is defined as $\Phi(x\ y\ \theta\ \lambda\ \dot{x}\ \dot{y})$ where x, y is the position, θ is the relative orientation, λ is a scale factor of the observed object in the image and $\dot{x}, \dot{y}$ denotes the relative motion of the object of interest in one frame to the subsequent frame. Let q be an a-priori known target model of the object of interest. Let $\textsc{Contour}(\Phi)$ be a function that is generated by a 3D projection of a known 3D shape of the object a contour model for all Φ. Our implementation



(a) inner and outer force (b) areas

Fig. 2. Principle of our tracking approach

defines this function for ball, goals, corner-posts and obstacles. The main idea of our approach is to keep the believed contour of an object of interest as close as possible to its background, instead of keeping it only close to the object's contour alone. This is illustrated in fig. 2(a), the arrows head for the two virtual forces that "push" the contour of the object to its correct position. The outer arrows indicate forces that maximise the likelihood of the statistics of the beliefs of the

objects. The inner arrows indicate forces, which result from the function that maximises the ratio of the ability to distinguish between object and background. This method prevents contours from collapsing because of low-contrast objects, like e.g. a white ball.

In practice, tracking of the exact contour of the object is difficult due to motion blur. Thus we track the region that is close to the object and its background region as shown in fig. 2(b). The real contour lies in the middle of both regions. We choose the $m-bin$ histogram for the feature space representation of the object region and the background region. Using histograms is not an optimal non-parametric density estimate method [5] but it suffices for our purposes. Both histograms are generated by function $h(x_i)$, which assigns the colour at location x_i to the corresponding histogram bin. In our experiments we use an illumination invariant colour space $r'g'I$ that was introduced in [12]. The conversion of RGB values is given by $(r', g', I) = (\frac{r}{g}, \frac{g}{b}, \frac{r+g+b}{3})$. Our experimental setup uses a $12 \times 12 \times 4$ bin histogram. The usage of a 4 bin illumination per 12×12 bin colour information provides a smaller sensitivity to illumination changes. Our experiments have shown that this also works by using only 16×16 colour bins without illumination information. However, in this case it becomes impossible to distinguish black from white objects. The colour distribution $p_{\Phi} = {y_u^{(u)}}_{u=1\ldots m}$ with the pose Φ is calculated as

$$p_{\Phi}^{(u)} = f \sum_{i=1}^{l} \delta[h(x_i) - u] \tag{1}$$

where l is the number of pixels in the area, δ is the Kronecker delta function and f is a normalisation constant ensuring $\sum_{u=1}^{m} p_{\Phi}^{(u)} = 1$. The probability distribution functions of both histograms are calculated separately. For the sake of simplicity we assume that all corresponding objects and background areas are known a-priori for all Φ. Note that both histogram bins are not binary e.g. we do not apply traditional segmentation. Next we need a similarity measure of $p_{object}(u)$ and $q_{object}(u)$ to enable tracking and target localisation. The bin-likelihood is calculated using the Bhattacharyya metric [5,13]:

$$\rho[p, q] = \sum_{u=1}^{m} \sqrt{p_{obj}(u) \cdot q_{obj}(u)} \tag{2}$$

In theory a perfect match will result in $\rho[p, q] = 1$. A non perfect match will result in a value between zero and one. To avoid certain observations, like for example a "white ball" on a "white wall", we can assume that the object is always distinguishable from the background. We extend (2) to

$$\rho[p, q] = \left\lfloor \left\lfloor \sum_{u=1}^{m} \sqrt{p_{obj}(u) \cdot q_{obj}(u)} - \sqrt{p_{bck}(u) \cdot q_{obj}(u)} \right\rfloor - \vartheta \right\rfloor \tag{3}$$

The background is extracted bin-wise instead of subtracting the summed likelihood. This leads to a higher stability in case of the background being alike to

the object scheme. In general, probabilistic frameworks output the "most believed" state, for example they will yield the "best fit". In practice this becomes often a problem when the object of interest is not visible in the image, e.g. due to partial or full occlusion, or when it is no longer in the image. In this case the next most believed state output will be considered as a false positive. In order to overcome that problem we apply thresholding using a lower limit $\vartheta \in \mathbb{R}$. Also, in the probabilistic framework of [10] a lower limit was used to detect the loss of an object.

Tracking whole regions yields relatively expensive computations, e.g. when we apply sequential Monte Carlo filters. Hence, we apply an approximation that uses structured *sample points* [8] near the contour, where the sample points can be considered as both, object and background. Using this technique allows us to have an adjustable trade-off between runtime and precision.

3 Probabilistic Object Tracking

In this section we consider the two co-operating memories of our approach. First we give a brief introduction to Particle Filters and the related *loss of diversity* problem in particle filtering. Next we explain the technical details of both memory models.

3.1 Particle Filters

Particle Filtering [10] or Monte Carlo [7] (MC) was developed to track objects in clutter. It is a Bayesian probabilistic method. The position of an object is represented by using a set of n weighted particles. Each particle contains a "believed" position with an assigned probability π. The pose of the object is estimated by using the observations as a function of the likelihood of believed poses/states while the particle filter attempts to maximise the likelihood of the beliefs. The usual MC algorithm works recursively in four different stages: (1) first the *prediction* stage of the motion model that is used to integrate the *actions* u to all particles e.g. the particles are simply moved. In the following stage (2) the observations are used to *update* the weight π of the particles. As next (3) the weight of all particles are normalised to one. Finally (4) the particles are *re-sampled* to get the posterior distribution. Technically, re-sampling discards particles with low weights and moves their weights to specific (random) particles with higher weights. In our implementation we move to the position of a new "offspring" particle with respect to the weight of the parent particle. For example a low weight of the parent particle will result in a high transformation.

3.2 Extended Particle Filtering (ePF)

Particle filters approximate probability density functions using a discrete set of n particles. Filters like the BOOTSTRAP algorithm typically approximate the density by maximising the probability of particles by re-sampling particles with

```
1: procedure extendParticleFilter
2:       // ...
3:     NormaliseImportanceWeights(π̃_t^(0), ..., π̃_t^(n))
4:       // Selection & Resampling step (Bayes Prediction Step)
5:       // generate m clusters from all particles
6:     c^(j)_{j=1...m} ← ObtainClusters(x̃^(i)_{0...t})
7:       // calculate the weight of the clusters
8:     ϖ^(j)_{j=1...m} ← NormaliseClusterWeights(ϖ'^(j)_{j=1...m})
9:     for l ← 1 to m do
10:        N'^(l)_t ← SelectCluster(c^(l))
11:        d ← Num(c^(l))    // Apply traditional resampling for each cluster..
12:    end for
13:      // ...
14: end procedure
```

Fig. 3. Extended Particle Filter, see [7] for the original algorithm

a "high" probability. The terms "high" and "low" are relative in the theory of probability density functions: The weights of all particles are usually normalised such that the overall sum is one. This means that particles with a "high" probability are considered more important than those with a relatively low probability. For example "high" ranked particles can generate more offspring in the re-sampling step.

The discrete approximation can lead to unwanted side effects caused by the discrete set of particles. For example, it can be the case when we apply particle filters to semi-bimodal distributions (one short dominant peak and a wide little peak): This can result in relatively many particles with a low probability and relatively few particles with a high probability. According to the probability function it can happen that low weighted particles gain a lot of attention of the filter compared to the few particles with a high probability, like e.g. the 'particle clinging ' effect when the system needs several time-steps to converge. With visual object tracking we face the problem that we have several areas with very low probabilities, e.g. if the background has a minimal similarity of the object. This is exactly the case with our object model (section 2). It is a relatively liberal model ensuring that lost objects are re-detected in short time. In the case of a lost object the particles are distributed randomly. Indeed the usage of a strict probability can reduce the required time to converge, however it has a high impact on the robustness of the system. In eq. (3) we introduced a lower limit to reduce the chance of this effect to occur. In literature this problem is sometimes referred as the "loss of diversity" (in particle filters) problem. In practise the probability density is unknown; hence we have to expect "particle clinging ". The "particle clinging " effect is also noticeable in the case of Monte Carlo localisation. With bi-modal distributions we face the problem that all particles will always converge to the highest probability [15] and may lose the object.

The main reason for the particle clinging effect is that we consider the particles piece-wise during the re-sampling step. We introduce a new *extended Particle*

filtering (ePF) method. It is based on the idea that we consider groups of particles (clusters) during re-sampling. This means that we treat clusters like ordinary particles. For example we assign each cluster a count of offspring particles. Each cluster generates its own offspring while we use the same method as in traditional methods, e.g. the traditional re-sampling step is applied to each cluster. We use an implementation based on the mean shift metric [2, pp 790]. This method builds clusters of the state space using a weighted kernel function and a kernel size. In contrast to the popular k-means clustering method the number of clusters is determined by the algorithm itself. Using the clusters we are able to obtain multiple states from one particle set by applying the MC estimation methods separately on each cluster. In practice 80% of the entire time only one cluster appears. In the worst case this extended method performs equally to the non-extended method.

3.3 Short-Term Memory

The short term memory deals with frame-to-frame changes of Φ_t in image sequences using the extended particle filters. We assume that the tracked object undergo translation or rotation in the image of the camera. Additionally we assume that only the object size changes while the shape remains unchanged. A straightforward model of such motion applied to the state is Φ_{t-1} to obtain Φ_t using:

$$\Phi_t = \Phi_{t-1} + \Delta_{mov} \tag{4}$$

To simplify matters we assume that the process noise is contained within Δ_{mov}. Based on the idea of the *Mass Inertia Model* we use the particle filtering method to estimate the translative motion. The relative motion is expressed by $\dot{x}$ and $\dot{y}$ through the translation:

$$\Phi_{xt} = \Phi_{xt-1} + \Phi_{\dot{x}t-1} \quad \Phi_{y_t} = \Phi_{y_{t-1}} + \Phi_{\dot{y}_{t-1}} \tag{5}$$

The theory of particle filters states that particles with a "good" proposal will survive in the sample set and populate in the resample set. The parameters $\dot{x}, \dot{y}$ are propagated in the resample step of the filtering process according to their prior weights. A "good" proposal will generate more but concentrated particles, while a "bad" proposal will generate fewer but highly varied particles. In theory the set will convert to "good" values after a few (> 3) iterations. So we obtain

$$\Phi_t = \Phi_{t-1} + \begin{pmatrix} \Phi_{\dot{x}t-1} \\ \Phi_{\dot{y}_{t-1}} \\ 1.1 - \pi'_{t-1} \\ 0 \\ 1.1 - \pi'_{t-1} \\ 1.1 - \pi'_{t-1} \end{pmatrix}^T \cdot \begin{pmatrix} 1 \\ 1 \\ \rho_\omega \\ 0 \\ \rho_{move} \\ \rho_{move} \end{pmatrix}^T \cdot \begin{pmatrix} 1 \\ 1 \\ \text{RND}() \\ 0 \\ \text{RND}() \\ \text{RND}() \end{pmatrix}^T + \Delta_{Noise} \tag{6}$$

where ρ_{move}, ρ_ω is the expected (maximum) relative movement and orientation change made in one time step, respectively. $\text{RND}() : [-1{:}1] \in \mathbb{R}$ is a non Gaussian

and non linear random function. π' represents the normalised weight ranging from $[0:1] \in \mathbb{R}$. Note that the (usual) π is normalised such that the sum of all π is one. To prevent a local minima we use 1.1 instead of 1.0 as normalisation value and we add uniform distributed noise Δ_{Noise} to the particles. Note that initially all parameters in Φ_t are set to zero.

3.4 Long-Term Memory

Although our particle filter is an efficient and robust visual object tracker, it only provides limited robustness to occlusions. In such a case the particle filter will converge to a uniform distribution of its particles. In order to keep objects for a "longer time" we use a well-tuned "Multiple Hypothesis Tracking"(MHT) filter. We apply a MHT algorithm that assigns an ID to the observed tracks of the particle filter. The original implementation given in [14], later extended in [6], provides already a full framework to suffice our needs. It is able to cope with missing measurements and predicts the motion of a-priori known unobserved objects. In our experimental setup, an unobserved object is "forgotten" after 2 seconds. Note that the original implementation is bound to linear/Gaussian systems.

We use a straightforward technique to give a feedback to the short-term memory: We add b additional particles to the particle filter using the positions of the predicted hypothesis of missing objects. The position of additional particles (per hypothesis) depends on its "age", e.g. the count of iterations until the last update step to a found observation. An older "age" will result in a larger spread of the additional particles. Note that the MHT filter is used only as a "long term memory" e.g. its only feeding "hints" to the short-term memory. The estimation of the believed state is done in the short-term memory, e.g. the most dominant observation is output. In the case of visually gone objects these "hints" are ignored by the particle filtering.

3.5 Ego-Motion Compensation

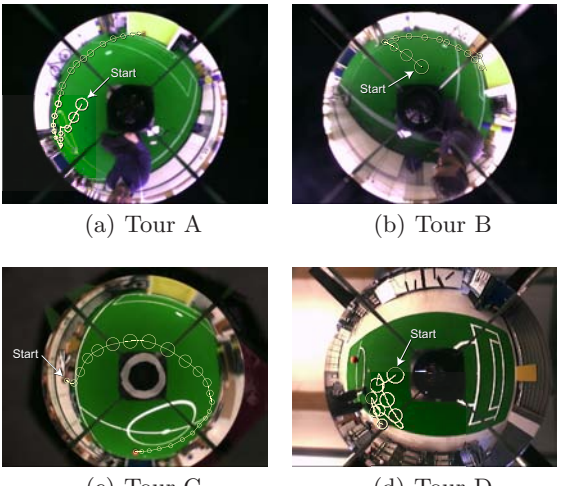
The main idea of ego-motion compensation is to apply the inverse motion of the robot to the tracker. In our experimental setup we use the self-localisation of the robot system. We use a simple heuristic to apply the *Ego-Motion Compensation* to the two tracker sensors: First we calculate the relative movement of the robot by using the difference of the current and previous poses, like e.g. the pose obtained from the current and previous iteration. We use the robot's pose probability as weight factor for the observed motion, which prevents "artificial motion" caused by non explicit poses, like for example during the global localisation. The weighted deltas are applied to the particles and the MHT hypothesis by shifting them. Note that the particles are only rotated. We do not shift the particles translatively because it can lead to unwanted behaviours. The MHT hypotheses are shifted translatively and rotatory according to their deltas. The motion models of both trackers remain untouched since we assume that the object itself moves (approaching the robot).

4 Experimental Results

In this section we will evaluate our approach and compare it to standard tracking methods like the *Mean Shift* tracker proposed by [5] and the *Colour-Based probabilistic tracking* proposed by [7,13]. The *Colour-Based probabilistic*approach is closely related to ours while *Mean Shift* differs from it. The *Mean Shift* is usually used in combination with a Kalman filter.

4.1 Test Environment

We use test sequences from the RoboCup Middle Size context to measure the performance. Our robot is equipped with an omni-directional camera system (omni-vision) which is the only sensor of the system. These kind of camera systems lead to the effect that *ego-motion* is more intense in the image as in ordinary pan and tilt setups.



(a) Tour A (b) Tour B

(c) Tour C (d) Tour D

Fig. 4. Sample Tours used for Evaluation, the unit of the values are frames

First, tour "A" tests the object tracking performance over a large range which is a typical situation in a RoboCup tournament: The ball can be far away from its observer while it is possible that the object moves rapidly towards the observer within short time. In tour "B" we assess the 'trackers' performance in case of highly dynamic object movements. The ball is pushed to the centre of the field and shot (very fast) towards the yellow goal. The shots are repeated three times and the ball bounces back. The objective of tour "C" is to test the stability w.r.t. ego motion which represents a typical behaviour of a RoboCup robot in the role of a striker. Tour "D" tests the stability of the trackers w.r.t. a background highly

Table 1. Accuracy of the trackers, the unit of all values is pixel

Tour	Mean Shift		Mean Shift + KL		MC Tracker		our approach	
	Avr.	Max	Avr.	Max	Avr.	Max	Avr.	Max
Tour A	21.125	168.848	5.510	48.664	40.253	348.319	4.651	62.047
Tour B	109.807	291.239	74.895	223.243	42.254	290.805	4.883	115.242
Tour C	153.664	350.957	168.434	513.469	26.951	238.484	9.273	221.215
Tour D	87.963	310.499	7.950	64.315	21.350	215.758	5.438	63.523

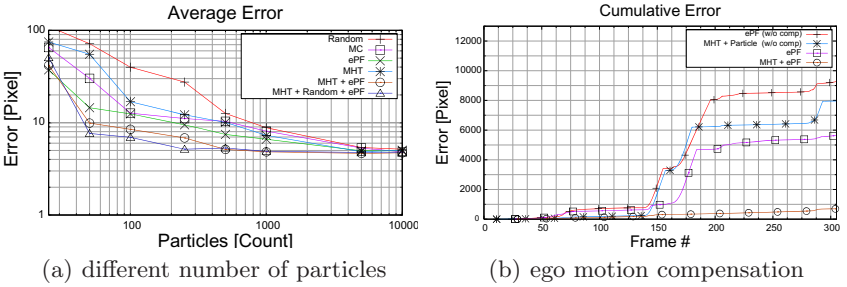
similar to the object of interest. Note that we use an ordinary white soccer ball which contrasts the RoboCup rule-set (2006).

4.2 Results

We use the following settings to parameterise our tracker: 200 particles for the *extended Particle Filter* module, 50 "hint" particles for the MHT module and 50 random particles. During the test runs the *ego-motion compensation* is not used. For the other contestant we use values that were proposed by [5,11]: an ellipsoid area is used for the *Mean Shift* and *MC*. We measure the Euclidean error $e1$ of the believed pose of the tracker with respect to the ground truth position. In tours with high dynamics the *MC* tracker performance is better than the *Mean Shift* tracker. Results are vice versa in tours with low dynamics. One reason for this is the incapability of a Kalman filter to adapt to non-linear/non-Gaussian dynamics. In tour C the Kalman filter degrades even the performance of the *Mean Shift* tracker. Altogether we see that our approach shows the lowest average error of all contestants.

4.3 Benchmarking

First we analyse the influences of the number of particles on the performance of different configurations of the tracker, see fig. 5(a). "MC" denotes a tracker using traditional Monte Carlo filtering. All configurations converge while the number



(a) different number of particles (b) ego motion compensation

Fig. 5. Using different configurations of our approach

of particles increases. The *MC* tracker performs better than the MHT tracker. In cased where MHT loses the object, all particles are set randomly to recover the position of the object. That's why MHT converges with an increasing number of particles. The extended Particle Filter method performs better than the traditional Monte Carlo. In fact the combination of MHT, random particles and extended Particle Filter shows the best results. The main reason for the improvement is that the MHT filters adapt better to "long term" motion (of the robot and objects) such as the extended Particle Filter, while the MHT filter fails to adapt to "short term" motion (e.g. if the ball is shot or is bounces back). It is also shown in [16] that random particles can improve the performance of particle filters. Next we consider the influence of the *ego motion compensation* on the accuracy of the *hybrid tracker*. Figure 5(b) shows the cumulative error of tour C using different configurations. Similar to fig. 5(a) an extended Particle Filter can be improved using additional particles resulting from the MHT "long term" memory. We see that *Ego motion compensation* significantly improves the performance. Turnings of the robot have only a slight influence on the performance.

5 Discussion

We showed that our new tracking technique can robustly track objects with high and low dynamics. Moreover we showed that the usage of our extended Particle Filter yields better performance compared to traditional Monte Carlo Filters. It allows us to obtain multiple (disjoint) independent states. The major drawback of our implementation is that we have to assume a *kernel size* for the object in the state space. The usage of too high or too low values can lead to multiple observations of the same object which degrades the performance to the level of traditional Monte Carlo filters. [4] proposed a parameter-free version of the *Mean shift* algorithm by analysing the density of the state space. We cannot apply this method because we use too few particles (≈ 1000 particles are needed). One advantage of our approach is that it only requires a minimum of initial knowledge about an object's motion dynamics since we assume that an object can move in any direction.

Furthermore we faced the problem of the dynamic scale adoption. In our approach we propagate these parameters using particles, where [5] applies a heuristic by using the best match probing the scales $\lambda, \lambda 0.9, \lambda 1.1$ in every iteration. Both solutions are sub-optimal and can lead to inaccuracy in case of poor illuminated environments. In fact the inaccuracy is caused by the false size adaptation which leads to a "shift" of the position of the believed track. See [5,13] for more details. In our approach we observed two bottlenecks: (1) the generation and evaluation of the colour scheme candidates and (2) the generation of the sample points of the approximated shape model. The computational time on the evaluation of the colour scheme highly depends on the number of used bins of the sensor model.

The *Mean Shift* is the fastest method of all tested trackers. This explains why it became popular in the visual tracking community. Theoretically the runtime of our approach and the *MC* tracker is identical. Practically m of our approach

Table 2. Computational complexity

	Mean Shift	MC Tracker	our Approach
Complexity	$O(m \log m)$	$O(nm \log m)$	$O(nm \log m)$
Average Runtime	$< 1ms$	$\approx 20ms$	Ball:$\approx 6ms$, Goal $\approx 11ms$
typical n	-	150	300
typical m	30x30	30x30	16x8

is much smaller than MC due to the fact that we use sample pixels instead of the pixel areas. The runtime was measured on a 1.1GHz PentiumM notebook.

References

1. Bar-Shalom, Y., Fortmann, T.: Tracking and data association. Mathematics in science and engineering, 1st edn., vol. 179, Academic Press Inc., London (1988)
2. Cheng, Y.: Mean shift, mode seeking, and clustering. IEEE Transactions on Pattern Analysis and Machine Intelligence 17, 790–799 (1995)
3. Collins, R., Liu, Y., Leordeanu, M.: On-line selection of discriminative tracking features. IEEE Transaction on PAMI 27(10), 1631–1643 (2005)
4. Comaniciu, D., Meer, P.: Mean shift: A robust approach toward feature space analysis. IEEE Transaction on PAMI 24, 603–619 (2002)
5. Comaniciu, D., Ramesh, V., Meer, P.: Kernel-based object tracking. IEEE Transaction on PAMI 25, 564–575 (2003)
6. Cox, I., Hingorani, S.: An efficient implementation and evaluation of reid's mht algorithm for visual tracking. In: ICPR94, pp. 437–442 (1994)
7. Doucet, A., Freitag, N., Gordon, N.: Sequential Monte Carlo Methods in Parctise, ch. 2, pp. 4–16. Springer, New York (2001)
8. Hanek, R., Schmitt, T., Buck, S., Beetz, M.: Towards robocup without color labeling. In: RoboCup International Symposium 2002 (2002)
9. Hundelshausen, F., Rojas, R.: Tracking regions. In: Polani, D., Browning, B., Bonarini, A., Yoshida, K. (eds.) RoboCup 2003. LNCS (LNAI), vol. 3020, Springer, Heidelberg (2004)
10. Isard, M., Blake, A.: Condensation – conditional density propagation for visual tracking. International Journal of Computer Vision 29(1), 5–28 (1998)
11. Nummiaro, K., Koller-Meier, E., Gool, L.: An adaptive color-based particle filter. Image and Vision Computing 21(1), 99–110 (2003)
12. Olufs, S.: Realtime color-segmentaion of fast moving objects (in german). Master's thesis, University of Applied Sciences Bonn-Rhein-Sieg (2002)
13. Prez, P., Hue, C., Vermaak, J., Gangnet, M.: Color-based probabilistic tracking. In: Heyden, A., Sparr, G., Nielsen, M., Johansen, P. (eds.) ECCV 2002. LNCS, vol. 2350, pp. 661–675. Springer, Heidelberg (2002)
14. Reid, D.: An algorithm for tracking multiple targets. IEEE Transaction on Automatic Control 24(6), 843–854 (1979)
15. Schulz, D., Burgard, W., Fox, D., Cremers, A.: People tracking with a mobile robot using sample-based joint probabilistic data association filters. Journal of Robotics Research (IJRR) (2003)
16. Thrun, S., Fox, D., Burgard, W., Dellaert, F.: Robust monte carlo localization for mobile robots. Artificial Intelligence 128(1-2), 99–141 (2000)