REAL-TIME MULTI-OBJECT TRACKING WITH FEW PARTICLES
A Parallel Extension of MCMC Algorithm

François Bardet, Thierry Chateau and Datta Ramadasan
LASMEA, Université Blaise Pascal, 24 avenue des Landais, F-63177 Aubière cedex, France

Keywords: Multi-object tracking, MCMC particle filter, Parallel computing, Real time.

Abstract: This paper addresses real-time automatic tracking and labeling of a variable number of generic objects, using one or more static cameras. The multi-object configuration is tracked through a Markov Chain Monte-Carlo Particle Filter (MCMC PF) method. As this method sequentially processes particles, it cannot be speeded up by parallel computing allowed by multi-core processing units. As a main contribution, we propose in this paper an extended MCMC PF algorithm, benefiting from parallel computing, and we show that this strategy improves tracking operation. This paper also addresses object tracking involving occlusions, deep scale and appearance changes: we propose a global observation function allowing to fairly track far objects as well as close objects. Experiment results are shown and discussed on pedestrian and on vehicle tracking sequences.

1 INTRODUCTION

Real-time visual tracking of a variable number of objects is of high interest for several applications such as visual surveillance of people, animals or vehicles. In the recent years, several works addressed these fields, showing interesting results. Among others: tracking up to 4 pedestrians involving occlusions (Isard and MacCormick, 2001), multiple vehicle tracking from a low elevation camera (yielding occlusions) (Kanhere and Birchfield, 2005), tracking up to 20 ants from a top view (Khan et al., 2005). The work presented below is closely inspired by the latter, extended to a generic tracker where deep occlusions frequently occur, and where object and background appearance may change. We also want the tracker to operate upon any kind of opaque generic objects (modeled by a cuboid), requiring no ad-hoc features but only object dimensions and its dynamics model if any (in case of a vehicle). Designing a multi-object tracker complying with these requirements is still challenging.

Particle Filters belong to the class of Monte-Carlo recursive Bayesian filters, and is of popular use in the field of object tracking, because Particle Filters can cope with non-linearities and multi-modalities induced by occlusions and background clutter. As a sampled method, Particle Filters require a way to smartly choose and propagate the samples (the "particles") over time. As the system state may evolve at each time step, samples must be moved adaptatively towards the most informing regions of the state space. This is a sequential adaptive resampling method known as SIR algorithm (Sample Importance Resampling) (M. Isard and A. Blake, 1998). A monocular multi-object tracker based on it was proposed (Isard and MacCormick, 2001), and many other works followed. The drawback of SIR, shown by many authors (Isard and MacCormick, 2001; Smith, 2007), is that it can't deal with high-dimensional state-spaces, because the number of required particles grows as an exponential of the state-space dimension. Thus, a SIR Particle Filter can't track more than 2 or 3 objects. Partitionned Particle Filter (MacCormick and Blake, 1999) has been proposed to overcome this limitation, but it processes objects by order, yielding to unfairly tracking them (Smith, 2007). An off-line non-sequential approach, when real-time is not required, uses a Markov Chain Monte-Carlo state space exploration to associate data in order to track objects over a whole sequence (Yu et al., 2007). For real-time tracking, Markov Chain Monte-Carlo Particle Filters (MCMC PF) have been shown to successfully track 2 to 4 pedestrians (Smith, 2007). More objects (up to 20 ants) (Khan et al., 2005) have successfully been tracked in a case where occlusions are avoided (top view of ants walking on a planar ground). Both emphasized the benefit of MCMC PF: the required number of particles is only a linear function of the number of tracked objects, when they do not interact. More computation is only required in case objects interact

Bardet F., Chateau T. and Ramadasan D.
REAL-TIME MULTI-OBJECT TRACKING WITH FEW PARTICLES - A Parallel Extension of MCMC Algorithm.
DOI: 10.5220/0001779104570464
In *Proceedings of the Fourth International Conference on Computer Vision Theory and Applications (VISIGRAPP 2009)*, page
ISBN: 978-989-8111-69-2

(i.e. occlude). In this case, real time tracking of several objects yet is a challenge. As a main contribution, we propose in this paper an extended MCMC PF algorithm, benefiting from parallel computing, and we show that this strategy improves tracking operation. To address object occlusions and deep scale changes, we also propose in this paper a global observation function allowing to fairly track far objects as well as close objects. Experiment results are shown and discussed on pedestrian and on vehicle tracking sequences. This paper is organised as follows: in section 2, we develop the MCMC PF method for tracking a variable number of objects. In section 3, we extend the MCMC PF method in order to benefit of multi-core processing units. In section 4, we describe the observation function, focusing on their independance to scale change. In section 5, tracking results are demonstrated and discussed with a focus on real-time capabilities.

2 MULTI-OBJECT MCMC PF

Let $p(\mathbf{X}_t|\mathbf{Z}_{1:t})$ denote the posterior probability density for a system state to be $\mathbf{X}_t$ at time t, knowing an observation sequence $\mathbf{Z}_{1:t}$. Particle Filters propagate a number N of particles over time, to approximate $p(\mathbf{X}_t|\mathbf{Z}_{1:t})$ as a sum of Dirac functions, such that: $p(\mathbf{X}_t|\mathbf{Z}_{1:t}) \approx \frac{1}{N}\sum_{n=1}^{N}\delta(\mathbf{X}_t - \mathbf{X}_t^n)$ where $\mathbf{X}_t^n$ denotes the n-th state sample at time t. In MCMC Particle Filters, these samples are drawn iteratively, through a first order Markov process.

2.1 State Space

In object tracking the state encodes the configuration of the perceptible objects: $\mathbf{X}_t^n = \{I_t^n, \mathbf{x}_{t,i}^n\}$, $i \in \{1,...,I_t^n\}$, where I_t^n is the number of visible objects for hypothesis n at time t, $n \in \{1,...,N\}$ where N is the number of iterations, and $\mathbf{x}_{t,i}^n$ is a vector encoding the state of object i, such that $\mathbf{x}_{t,i}^n = \{\mathbf{p}_{t,i}^n, \mathbf{v}_{t,i}^n, \mathbf{s}_{t,i}^n, \mathbf{a}_{t,i}^n\}$. The object i position at iteration n is described by $\mathbf{p}_{t,i}^n$, a 3-component vector including the 2D ground projection of the object center of gravity, and its orientation angle. The ground is assumed to be planar. The object velocity is described by $\mathbf{v}_{t,i}^n$, a 2-component vector including the velocity magnitude and orientation. Its shape is described by $\mathbf{s}_{t,i}^n$, a 3-component vector including width, length and height of a cuboid approximating the object shape. $\mathbf{a}_{t,i}^n$ denotes its appearance vector, helping to maintain object identity. We use the color model proposed in (P. Perez et al., 2002): we convert our images to a hue-saturation-value color space, then pixels with sufficient value

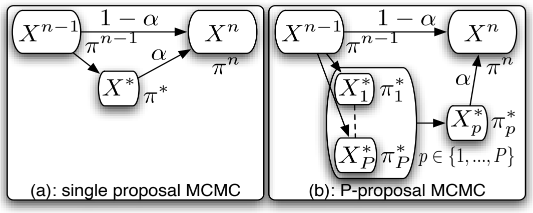

Figure 1: (a): one step of algorithm 1 produces a unique proposal $\mathbf{X}^*$, accepted as next state $\mathbf{X}^n$ with probability α. If rejected then $\mathbf{X}^n = \mathbf{X}^{n-1}$. (b): one step of algorithm 2 produces P proposals $\mathbf{X}_p^*$. One of them is drawn by importance sampling according to weights π_p^*, then accepted as next state $\mathbf{X}^n$ with probability α. If rejected then $\mathbf{X}^n = \mathbf{X}^{n-1}$.

and saturation feed an unmarginalized hue-saturation histogram, other pixels a value histogram. Both are concatenated to build a color model whose benefit is a lesser sensitivity to illumination changes.

2.2 MCMC PF for Multi-Object Tracking

MCMC PF for tracking a variable number of objects was introduced in (Khan et al., 2005), and is described in algorithm 1. Omitting time t for sake of simplicity, figure 1-a focuses on the markovian transition from particle $\mathbf{X}^{n-1}$ to particle $\mathbf{X}^n$ via a unique new proposal $\mathbf{X}^*$, which may be accepted with probability α (defined in algorithm 1). If refused then $\mathbf{X}^n$ is a duplicate of $\mathbf{X}^{n-1}$. This is the Metropolis-Hastings acceptance rule (MacKay, 2003). Please note that computing proposal $\mathbf{X}^*$ likelihood π^* is by far the heaviest step in the algorithm, as it involves image wide computations.

2.3 Marginalized Proposal Moves

Basic Particle Filters (i.e. those which draw new samples by moving jointly along all the dimensions) can't cope with a high dimension state space, because the required number of samples grows as an exponential of the space dimension, as focused in (Smith, 2007). The best issue to this problem would be to process Metropolis-Hastings algorithm with a proposal move on only one randomly chosen dimension at each iteration. We cannot choose this optimal solution, because the components of $\mathbf{p}_{t,i}^n$ and $\mathbf{s}_{t,i}^n$ are not independantly observable. For that reason, we chose a midway solution: the transition from state hypothesis $\mathbf{X}$ to the next $\mathbf{X}^*$, is conditionned by a proposal density $q(\mathbf{X}^*|\mathbf{X})$, allowing changes along all the dimensions of one randomly chosen object at a time, within its own state subspace. This solution was also chosen by other authors (Khan et al., 2005; Smith, 2007).

Table 1: Prior object dimensions.

pedestrian	**mini**	**maxi**
length (m)	0.15	0.3
width (m)	0.25	0.45
height (m)	1.2	2.0
velocity magnitude ($m.s^{-1}$)	1	2
velocity angle (deg)	0	360
vehicle	**mini**	**maxi**
length (m)	3.5	5.5
width (m)	1.4	2
height (m)	1.4	2.2
velocity magnitude ($m.s^{-1}$)	10	30
velocity angle (deg)	-30	+30

2.4 Variable Number of Objects

To allow the number of objects to change, authors introduced RJMCMC (Reversible Jump Markov Chain Monte Carlo) (Green, 1995). As the number of visible objects may change, the state space dimension also may change, the state may thus "jump" from a subspace to another one of larger or smaller dimension if a new object enters or leaves the scene. To prevent the search chain from getting stuck in a local minimum, the jumps between subspaces must be reversible. For that reason, authors proposed the pair of discrete reversible moves $\{enter, leave\}$ (Khan et al., 2005; Smith, 2007). To improve time consistency, object *leave* proposals are driven by its life expectancy, a continuous variable updated at each iteration as a function of how the particle object matches the observation. The *object update* move is self-reversible as it is a continuous move. The whole move set will be denoted as $M = \{enter, leave, object\ update\}$.

2.5 Proposal Moves

Enter: Propose a new joint configuration $\mathbf{X}^* = \{\mathbf{X}^{n-1}, \mathbf{x}_{I^{n-1}+1}\}$, adding a new object $\mathbf{x}_{I^{n-1}+1}$ to the previous joint configuration $\mathbf{X}^{n-1}$, where I^{n-1} is the number of objects hypothesized by $\mathbf{X}^{n-1}$. Entering object is given an initial life expectancy, and its priors are given in table 1. The search process jumps to a higher dimension state subspace.

Leave: The reverse move proposes to withdraw object i from $\mathbf{X^{n-1}}$: propose $\mathbf{X}^* = \{\mathbf{X}^{n-1} \setminus \mathbf{x}_i^{n-1}\}$, $i \in \{1, ..., I^{n-1}\}$, I^{n-1} is hypothesis $\mathbf{X}^{n-1}$ object number, and $\{s \setminus e\}$ is set s without element e. The search process jumps to a lower dimension state subspace.

Object Update: At each iteration n of time t Markov chain, we randomly choose an object j to be updated among particle $\mathbf{X}_t^n$. We randomly choose a time $t-1$ particle $\mathbf{X}_{t-1}^r$, $r \in \{1, ..., N\}$, also involving object j, within the input set:$\{\mathbf{X}_{t-1}^n\}_{n=1}^N$. From this object j instance $\mathbf{x}_{t-1}^{j,n}$, we draw a position, velocity and shape proposal, according to equations 1, 2, 3:

$$q(\mathbf{p}^*|\mathbf{p}_{t-1}^r) = \mathcal{N}(\mathbf{p}_{t-1}^r, \sigma_p^2 I_2) \quad (1)$$

$$q(\mathbf{v}^*|\mathbf{v}_{t-1}^r) = \mathcal{N}(\mathbf{v}_{t-1}^r, diag\left(\left[\sigma_m^2, \sigma_a^2\right]\right)) \quad (2)$$

$$q(\mathbf{s}^*|\mathbf{s}_{t-1}^r) = \mathcal{N}(\mathbf{s}_{t-1}^r, \sigma_s^2 I_3) \quad (3)$$

where σ_p is the position standard deviation, σ_m and σ_a are the respective velocity magnitude and orientation standard deviations, σ_s is the shape standard deviation, I_d is the d-dimension identity matrix.

Algorithm 1 MCMC Particle Filter.

```
Input: particle set at time $t-1$: $\{\mathbf{X}_{t-1}^n\}_{n=1}^N$
Initialize the chain: $\mathbf{X}_t^0 = \mathbf{X}_{t-1}^r$ $r \in \{1, ..., N\}$.
- Compute its likelihood: $\pi_t^0 \propto P(Z_t|\mathbf{X}_t^0)$.
for $i = 1$ to $N + N_B$ do
    - Randomly draw a move $m$ from move set $M$.
    if $m == object\ update$ then
        - Randomly choose object $j$ in particle $\mathbf{X}_t^{i-1}$.
        - Randomly choose $\mathbf{X}_{t-1}^n$, a particle involving object $j$: $\mathbf{x}_{t-1}^{j,n}$ from the input set:$\{\mathbf{X}_{t-1}^n\}_{n=1}^N$.
        - Apply dynamics to object $\mathbf{x}_{t-1}^{j,n}$: draw a sample $\mathbf{x}_t^{j*}$ from the evolution density $q(\mathbf{x}^*|\mathbf{x}_{t-1}^{j,n})$.
        - Build $\mathbf{X}^*$ replacing $\mathbf{x}_t^{j,i-1}$ by $\mathbf{x}_t^{j*}$ as object $j$.
    else
        - Propose an $enter$: $\mathbf{X}^* = \{\mathbf{X}_t^{i-1}, \mathbf{x}_{I^{i-1}+1}\}$
    end if
    - Compute its likelihood: $\pi^* \propto P(Z_t|\mathbf{X}^*)$.
    - Compute the acceptance ratio :
        $\alpha = min\left(1, \frac{\pi^* q(\mathbf{X}_t^{i-1}|\mathbf{X}^*)}{\pi_t^{i-1} q(\mathbf{X}^*|\mathbf{X}_t^{i-1})}\right)$
    - Add $\mathbf{X}_t^i = \mathbf{X}^*$ to the chain with probability $\alpha$, or $\mathbf{X}_t^i = \mathbf{X}_t^{i-1}$ with probability $1-\alpha$.
    - Update object $j$ life expectancy.
end for
Discard the $N_B$ first samples of the chain (burn-in).
Output: particle set at time $t$: $\{\mathbf{X}_t^n\}_{n=N_B+1,...,N_B+N}$
```

3 PARALLEL COMPUTING WITHIN MCMC FRAMEWORK

Multi-processor systems and multi-core processing units are now widely used. A multi-core unit can simultaneously process multiple independant tasks, so

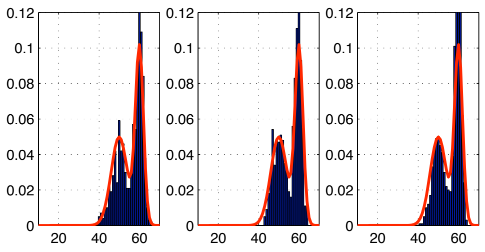

Figure 2: Sampling a monodimensional density function (red solid line) on $[0,100]$ range, with a MCMC sampler (histogram of particles in blue). Left: single proposal, center: dual proposal, right: 4-proposal. $N = 1000$ iterations, with iteration n proposal density: $\mathcal{N}(x^{n-1}, 5^2)$.

the global power benefit is conditionned to the possibility to balance computation loads between cores. In algorithm 1, the proposal likelihood computation required is by far the heaviest step in terms of computation costs, as it involves image wide computations (they will be detailed in section 4). Thus it is the best candidate task to be parallelized. Unfortunately, this task is required at every iteration of the chain, and its result is used to decide whether the proposal should be accepted or not as a new sample. Thus algorithm 1 cannot be parallelized as is.

In order to get benefit from parallel processors, we propose an extended version of algorithm 1, simultaneously generating mutiple state proposals and assessing their likelihoods. Algorithm 2 summerizes the operations. We denote it $MCMC^P PF$, where the superscript ^{P} is the number of processing cores on the machine processing unit. It differs from algorithm 1 in the P time replication of the proposal and evaluation mechanism, and in the additional importance sampling step that comes subsequently. Omitting time t for sake of simplicity, figure 1-b illustrates one step of algorithm 2, a markovian transition from particle $\mathbf{X}^{n-1}$ to particle $\mathbf{X}^n$ via P new proposals $\mathbf{X}^*_p$, $p \in \{1,\dots,P\}$. Each processing core receives one of these state proposals, evaluates its likelihood $\pi^*_p \propto P(Z_t|\mathbf{X}^*_p)$ and returns it to the main process. One of them is then drawn by importance sampling among all the parallel proposals and their associated likelihoods π^*_p. It is accepted as next state $\mathbf{X}^n$ with probability α. If rejected then $\mathbf{X}^n$ is the duplicate of $\mathbf{X}^{n-1}$.

Figure 2 shows that single and multi-proposal MCMC samplers similarly estimate any target density. Figure 3 shows that our $MCMC^P PF$ behaves similarly regardless to the value of P. As a filter, it smoothly starts moving from $t = 1$ towards the new probability density. It also shows that multiple proposal filters converge faster, allowing to reduce the chain length.

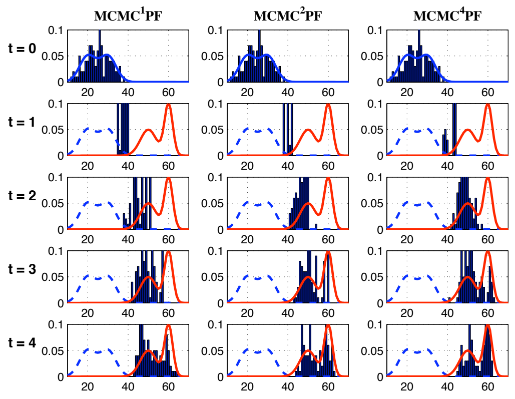

Figure 3: Filtering a monodimensional time-evolving density function (red solid line) on $[0,100]$ range, with a $MCMC^P PF$filter (histogram of particles in blue). Left: single proposal $P = 1$, center column: dual proposal $P = 2$, right: 4-proposal $P = 4$. Steady state until t=0 (upper row). At time t=1 (second row), density dramatically changes then remains steady until t=4 (bottom row). $N = 100$ particles with particle n proposal density: $\mathcal{N}(x^{n-1}_{t-1}, 5^2)$.

4 OBSERVATION FUNCTION

4.1 Background Model

Though only one camera is used in many surveillance applications, more cameras may be required, either to increase the trajectory estimate accuracy, or to increase the measurement range. For that reason, our measurement function can merge data from one or more cameras. Assuming still cameras and background, we first process a simple background subtraction on each frame. Then each pixel is classified as background or foreground, and we compute a binary foreground image according to equation 4:

$$\mathbf{I}_F(g,c) = \begin{cases} 1 \text{ if } p(\mathbf{z}_{g,c}|l_{g,c} = F) > p(\mathbf{z}_{g,c}|l_{g,c} = B) \\ 0 \text{ otherwise} \end{cases} \tag{4}$$

where $\mathbf{z}_{g,c}$ denotes the pixel value at location g of the frame sent by camera number c at time t, $\forall g \in \{1,\dots,G\}$ with G the number of processed pixels of the frame, $\forall c \in \{1,\dots,C\}$, with C the number of cameras in use, and $l_{g,c} \in \{B,F\}$ is a label assigned to the pixel at location g, according to whether the state hypothesis $\mathbf{X}$ assumes that this pixel is illuminated either by the Background B, or by a Foreground object F.

4.2 Global Observation Likelihood

Each object hypothesized by state $\mathbf{X}$ is modeled as a cuboid with shape defined by $\mathbf{s}^n_{t,i}$ as defined in section

Algorithm 2 *$MCMC^P$* Particle Filter.

```
Input: particle set at time $t-1$: $\{\mathbf{X}_{t-1}^n\}_{n=1}^N$
Initialize the chain: $\mathbf{X}_t^0 = \mathbf{X}_{t-1}^r$ $r \in \{1,...,N\}$, and its weight: $\pi_t^0) \propto P(Z_t|\mathbf{X}_t^0)$
for $i = 1$ to $N + N_B$ do
    for $p = 1$ to $P$ do
        - Randomly draw a move $m$ from move set $M$
        if $m == object\ update$ then
            - Randomly choose object $j_p$ from $\mathbf{X}_t^{i-1}$.
            - Randomly choose a particle involving object $j_p$: $\mathbf{x}_{t-1}^{j_p,n}$ from the input set: $\{\mathbf{X}_{t-1}^n\}_{n=1}^N$.
            - Apply dynamics to object $\mathbf{x}_{t-1}^{j_p,n}$: draw $\mathbf{x}_t^{j_p*}$ from the evolution density $q(\mathbf{x}^*|\mathbf{x}_{t-1}^{j_p,n})$.
            - Build $\mathbf{X}_p^*$ replacing $\mathbf{x}_t^{j_p,i-1}$ by $\mathbf{x}_t^{j_p*}$.
        else
            - Propose an enter: $\mathbf{X}_p^* = \{\mathbf{X}_t^{i-1}, \mathbf{x}_{I^{i-1}+1}\}$
        end if
        - Compute its likelihood: $\pi_p^* \propto P(Z_t|\mathbf{X}_p^*)$
    end for
    -Sample one of the $P$ proposals by importance sampling among the weighted proposal set : $\{\mathbf{X}^*, \pi^*\} \propto \{\mathbf{X}_p^*, \pi_p^*\}_{p=1}^P$
    - Compute the acceptance ratio :
        $$\alpha = min\left(1, \frac{\pi^* q(\mathbf{X}_t^{i-1}|\mathbf{X}^*)}{\pi_t^{i-1} q(\mathbf{X}^*|\mathbf{X}_t^{i-1})}\right)$$
    - Add $\mathbf{X}_t^i = \mathbf{X}^*$ to the chain with probability $\alpha$, or $\mathbf{X}_t^i = \mathbf{X}_t^{i-1}$ with probability $1-\alpha$.
    - Update object $j$ life expectancy.
end for
Burn-in : discard the $N_B$ first samples of the chain.
Output: particle set at time $t$: $\{\mathbf{X}_t^n\}_{n=N_B+1,...,N_B+N}$
```

2.1. We compute the convex hull of its vertice projections into the camera images. We then compute for each camera c a binary hypothesis image $\mathbf{I}_H(\mathbf{X}, g, c)$, setting pixel g value to 1 if it is inside at least one of the said convex hulls, else 0. A similarity image is finally computed $\mathbf{I}_S(\mathbf{X}, g, c)$:

$$\mathbf{I}_S(\mathbf{X}, g, c) = \begin{cases} 1 \text{ if } \mathbf{I}_F(g,c) = \mathbf{I}_H(\mathbf{X}, g, c) \\ 0 \text{ otherwise} \end{cases} \tag{5}$$

$\forall g \in \{1,...,G\}, \forall c \in \{1,...,C\}$.

Tracking multiple vehicles or pedestrians in traffic involves deep scale changes due to projection. Thus a naive image similarity ratio (the number of 1's in $\mathbf{I}_S$ divided by its area) would be much more discriminant for close object moves, than for further object moves. This drawback is illustrated by figure 4, where each object is proposed a horizontal move of half its width. Almost every move of the smallest objects would be accepted without discrimination. To solve this problem, we propose an original object-area dependent global observation likelihood, ensuring a target acceptance ratio, independantly to the object area. This is made possible because algorithm 1 only moves one hypothesized object at a time. We compute a parameter denoted β_c $\forall c \in \{1,...,C\}$, such that a given relative loss of object coverage k_S leads to a target acceptance criterium α_t. Let us denote S the area of the foreground binary image $\mathbf{I}_F$, $S_{o,c}$ the moved object area in camera c image. In our experiment, we choose $k_S = 0.1$ and $\alpha_t = 0.5$, meaning that an object proposal covering only 90% of the previous hypothesized object, would lead to an acceptance ratio $\alpha_t = 0.5$, thus equiprobably accepting or refusing this proposal. We compute the global likelihood as:

$$\pi_{F,c}(\mathbf{X}) = \left(\frac{1}{S}\sum_{g\in\{1,...,G\}} \mathbf{I}_S(g,c)\right)^{\beta_c} \forall c \in \{1,...,C\} \tag{6}$$

where $\pi_{F,c}(\mathbf{X}) = 1$ if the hypothesis mask $\mathbf{I}_H(\mathbf{X})$ perfectly fits $\mathbf{I}_F$, and $\pi_{F,c}(\mathbf{X}) = \left(\frac{S-2S_{o,c}}{S}\right)^{\beta_c}$ if they do not intersect at all. The parameter β_c can be computed as:

$$\beta_c = \frac{log(\alpha_t)}{log(1 - 2k_S\frac{S_{o,c}}{S})} \tag{7}$$

Acceptance ratio no more depends on the object distance, as illustrated on figure 4. Remaining disparities are due to the object shapes which do not fit all the same the hypothesized rectangles. This observation function gives us the opportunity to adjust the Metropolis-Hastings acceptance rate, as it determines the search efficiency. Finally the global configuration $\mathbf{X}$ likelihood required to compute algorithms 1 or 2 acceptance rate, is calculated as:

$$\pi(\mathbf{X}) = \prod_{c=1}^{C} \pi_{F,c}(\mathbf{X}) \tag{8}$$

5 EXPERIMENTS AND RESULTS

5.1 Synthetic Data Experiments

Particle filters are suitable to estimate any kind of state and observation probability densities, without any gaussian hypothesis, as illustrated on figure 2. Nevertheless, in order to assess and compare algorithms 1 and 2 efficiencies, we choose to use multivariate gaussian probability densities in order to allow the use of a Kalman Filter as a reference, as it is known to be an optimal filter in such conditions. This test is based on fully synthetic data, as defined below:

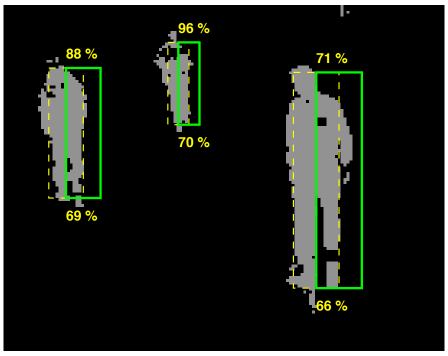

Figure 4: Foreground image and projected multi-object hypothesis. Dashed line: initial hypothesis. Solid line: proposal hypothesis. Their acceptance rates with a naive observation function are given on the upper side of the rectangles. Their acceptance rates with our observation function are given on the lower side of the rectangles.

- We want to estimate a 20-dimensional state with [0,1] range along each dimension.
- Suppose that previous measurements led our filters to the following system state estimate at time $t-1$: $\mathcal{N}(0.36 * I_{20}, 0.05^2 * I_{20})$, where I_{20} is the 20-dimensional identity matrix.
- The system dynamics prediction (or proposal) is modelled by: $\mathcal{N}(0, 0.1^2 * I_{20})$.
- The observation likelihood function at time t is $\mathcal{N}(0.4 * I_{20}, 0.05^2 * I_{20})$.

The chosen dimension is typical of tracking 5 cylindric objects, each of them being described by a 4-dimensional subspace: its 2-D position, its radius and its height. Figure 5 shows the clear benefit of increasing P: it allows either to shorten the chain given a target deviation, or to decrease the deviation given the length chain. Parallel evaluation of proposals also causes additional processing costs as some communication must occur between the processing cores in order to comply with the iterative structure of the algorithm. We use parallel processing through threads supplied by the Boost C++ Libraries[1]. One thread is run on each processing core. We used NT^2 C++ library[2]. Experiments are conducted on a 3GHz Intel E6850 Core 2 Duo processor PC, with 3,25Go RAM, running Linux Kubuntu 7.10. Table 2 shows that multiple proposal benefits outweigh the associated additional costs: dual proposal $MCMC^2PF$ performs faster than $MCMC^1PF$ as it requires less particles to reach the target.

[1]http://www.boost.org
[2]http://nt2.sourceforge.net

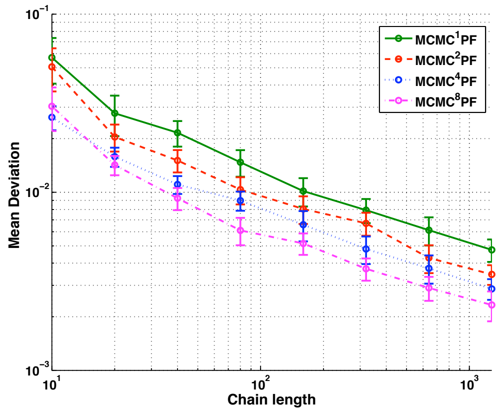

Figure 5: Average deviation of $MCMC^PPF$ estimated mean relative to Kalman Filter estimated mean, versus chain length for $P \in \{1,2,4,8\}$ parallel proposals. Measurement is a time-evolving 20-dimensional Gaussian distribution. Errorbar heights are twice the standard deviation.

Table 2: Tracking five 4-dimensional objects with a target deviation equal to 1% of the measurement range. The given times only include the tracker computation. Additional 10ms are required for foreground segmentation.

proposal	**chain length**	**time(ms)**	**fps**
single	150	30	25
dual	90	23	30

5.2 Video Experiments

We designed this tracker to be as generic as possible, in order to track several classes of objects, such as pedestrians or vehicles, in any scenery. For that purpose, pedestrians as well as vehicles obey the same constant velocity dynamics model. Only the values are object-specific (see table 4), and the vehicle bicycle dynamics model was not active in this experiment for sake of genericity (though it greatly improves vehicle tracking). For the same reason, we let objects *enter* or *leave* the scene without any location prior, and object velocity priors are very loose, as can be seen in table 1, allowing experiments in various sceneries. On a defined application, *enter* and *leave* locations may be more constrained or learnt online, as well as velocity angle and magnitude. It makes no doubt that these priors also will improve tracking. Figure 6 image #1 shows that no entering prior is used, allowing to start tracking at any time. After processing image #1, vehicle #1 is estimated to be present, others will appear on the following frames. On a first scenery, we track and label multiple vehicles on a highway video sequence from the traffic database used at the Statistical Visual Computing Lab

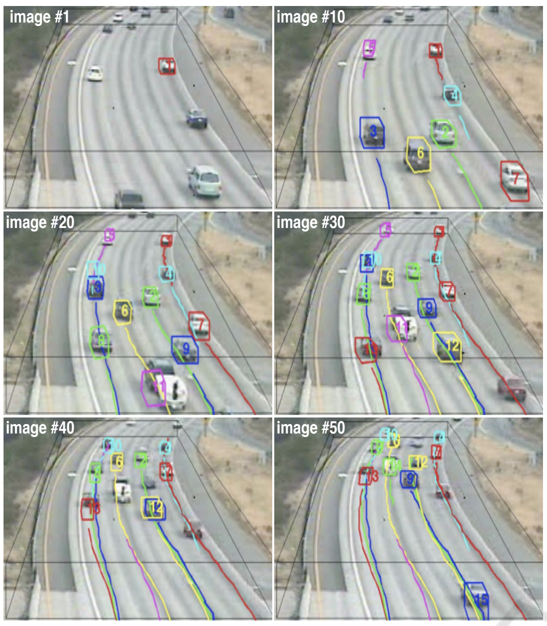

Figure 6: Highway sequence. Thin black lines define the limits of the 20*100 meters tracking area. Tracked vehicle estimated cuboids and their past trajectories are overplotted.

at UCSD[3], captured with a low-resolution (320*240), low frame rate (10 fps) surveillance webcam, and is typical of traffic surveillance applications. Figure 6 shows few frames from one of these sequences. On a second scenery, we track and label multiple pedestrians captured with a 640*480, 30 fps IP camera, downsampled to 320*240 (see figure 7). On both figures, overplotted thin black lines define the limits of the tracking area. When entering this area, a new object is labeled with a unique identifier, plotted with its estimated convex hull.

5.3 Multiple Vehicle Tracking

To evaluate the tracker performance, we choose a short (50 frames) but significant highway sequence, involving up to 12 vehicles (average number: 9.5) within a 20x100 meter tracking area. The tracker performance is measured through an error rate $\varepsilon = \sum_{r=1}^{R}\sum_{t=1}^{50}\frac{n_f}{n_v}$, where n_f denotes the number of tracking failures and n_v the number of target vehicles, R is the number of repetitions of the experiment, set to $R = 10$, yielding a total number of 4930 vehicle*frame to be tracked. Ground truth has been set by hand, frame by frame. Table 3 sums up error rate and frame rate, and shows that the dual proposal filter *MCMC*^{2}*PF* clearly outperforms the single proposal

[3]http://www.svcl.ucsd.edu/

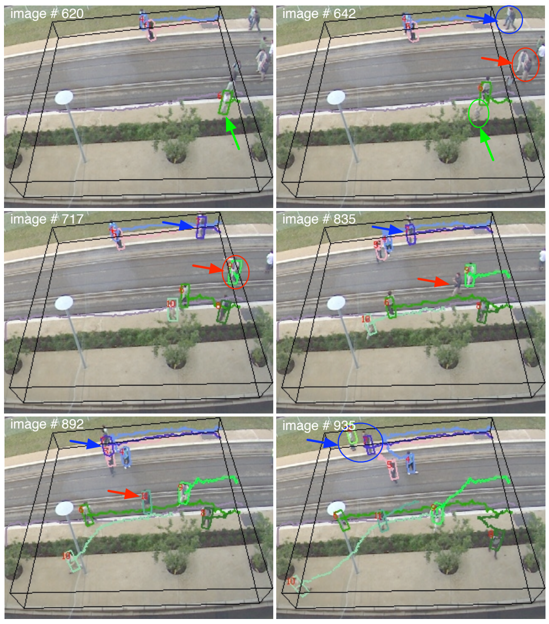

Figure 7: Pedestrian sequence. Thin black lines define the limits of the 12x10 meter tracking area. Tracked pedestrian estimated cuboids and their identity are overplotted.

Table 3: Vehicle tracking failure rate ε and frame rate.

proposal	particles	error ε	fps
single	150	0.27	27
dual	90	0.18	33
single	300	0.18	15
dual	180	0.13	19

on both criteria. For this experiment a constant velocity model has been used according to table 4 data.

5.4 Multiple Pedestrian Tracking

The tracker performance is assessed as defined in subsection 5.3, but over larger sequences (6000 frames). From 2 to 10 pedestrians are tracked on a 15x12 meter area. Table 5 results reinforce section 5.3 statements. Higher frame rate is due to the lower object average number (4.3 instead of 9.5).

Table 4: Object dynamics.

object	$\sigma_p(m)$	$\sigma_m(m/s)$	$\sigma_a(rad/s)$	$\sigma_s(m)$
human	0.4	0.1	0.3	0.05
vehicle	0.75	0.1	0.02	0.1

Table 5: Pedestrian tracking failure rate ε and frame rate.

proposal	particles	error ε	fps
single	150	0.17	31
dual	90	0.14	37
single	300	0.14	17
dual	180	0.13	19

5.5 Tracking Failures

Losing track of an object: mostly caused by poor foreground-background segmentation, as illustrated on figure 7: object target labeled as #6 (green arrow) in image #620 is lost in image #642 and its estimate shifts towards a new target. It will be tracked again at image #717, under a new label: #10. When many objects are being tracked, the tracker devotes too few iterations to an entering target, yielding coarse initialization of a new object onto this target, and sometimes missing it. This is shown on figure 6 by the red vehicle entering bottom right on image #30.
Double tracking of a unique target: on figure 6 image #20 a new vehicle #10 is superimposed onto a target already tracked by object #3. This error will be recovered between images #30 and #40.
Single tracking of two targets: on figure 7 image #642, two pedestrians (blue arrow) will enter while occluding each other. They are tracked as single object #7, and will be recovered at #935, as soon as there is evidence that two objects are present. This shows the benefit of the absence of enter location prior. Similarly #9 (red arrow) splits #835 (recovery delayed to image #892 due to poor foreground segmentation).

6 CONCLUSIONS AND FUTURE WORKS

We have presented a generic multi-object real-time automatic tracking system, using MCMC Particle Filter. We have proposed a Multi-Proposal MCMC Particle Filter (denoted $MCMC^P PF$) algorithm, allowing to compute in parallel P proposal likelihoods (the most computation consuming task), benefitting from the use of muti-core processing units. We have shown that dual proposal $MCMC^2 PF$ outperforms single proposal $MCMC^1 PF$, improving tracking while requiring less particles, thus yielding higher frame rate. Our synthetic data experiments allow to generalize this result to up to 8 parallel proposals, allowing to look forward to much improved tracking performance, especially to track a higher number of objects, typically 20 to 30 for highway surveillance. The global likelihood observation function allows to cope with occlusions and deep scale changes. Though now only tracking a single object class of objects, the ultimate goal of this research is to simultaneously track and classify several classes of objects, such as road users, including trucks, cycles and pedestrians, in order to analyze road users interactions. For that purpose, object model selection will be proposed within the MCMC framework, as a discrete random variable.

REFERENCES

Green, P. J. (1995). Reversible jump markov chain monte carlo computation and bayesian model determination. *Biometrika*, 4(82):711–732.

Isard, M. and MacCormick, J. (2001). Bramble: A bayesian multiple-blob tracker. In *Proc. Int. Conf. Computer Vision, vol. 2 34-41*.

Kanhere, N. K. Pundlik, S. J. and Birchfield, S. T. (2005). Vehicle segmentation and tracking from a low-angle off-axis camera. In *CVPR, Conference on Computer Vision and Pattern Recognition*, volume 2, pages 1152–1157.

Khan, Z., Balch, T., and Dellaert, F. (2005). Mcmc-based particle filtering for tracking a variable number of interacting targets. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27:1805 – 1918.

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

MacCormick, J. and Blake, A. (1999). A probabilistic exclusion principle for tracking multiple objects. In *Int. Conf. Computer Vision, 572-578*.

MacKay, D. (2003). *Information Theory, Inference, and Learning Algorithms*. Cambridge University Press.

P. Perez, C. Hue, J. Vermaak, and M. Gangnet (2002). Color-Based Probabilistic Tracking. In *Computer Vision ECCV 2002*, volume 1, pages 661–675.

Smith, K. (2007). *Bayesian Methods for Visual Multi-Object Tracking with Applications to Human Activity Recognition*. PhD thesis, EPFL, Lausanne, Suisse.

Yu, Q., Medioni, G., and Cohen, I. (2007). Multiple target tracking using spatio-temporal markov chain monte carlo data association. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1 – 8.