Technical report

Particle filtering for velocity estimation in image sequences

Mark Coates, Boris Oreshkin

Email: mark.coates@mcgill.ca, boris.oreshkin@mail.mcgill.ca

CONTENTS

I. Introduction

The problem of traffic monitoring and parameter estimation using camera networks has attracted a lot of attention during recent years. This is due to the fact that many large camera networks have been built along roadways for operator aided traffic monitoring. These camera networks supply traffic monitoring centers with large amounts of real-time data. This information source can be readily used to automatically extract traffic parameters, such as its intensity or velocity, without any additional costs involved just by using image processing techniques. However, in many practical situations this traffic estimation goal is not so easy to achieve as it might seem from the first glance. For example, data from cameras along the roads are often compressed to a very low resolution and low frame rate video before they are sent to traffic monitoring center. Thus data available to potential end users in the person of image processing and computer vision community is often of very low quality. This distinctive feature of the problem often makes it impossible to apply existing computer vision techniques to traffic estimation directly. Notwithstanding the potential problems outlined above, most of the work in the field of traffic estimation is made under the assumptions of high resolution and high frame rate video available for optical flow analysis and feature extraction. These methods mostly rely on geometrical methods for the extraction of traffic speed vectors and availability of good vehicle models that can be used to detect and even identify cars on the road. All these features move aforementioned methods down into the class of deterministic algorithms that can deal only with cases when data are not noisy, highly informative and objects of interest transform into sensor data obeying almost completely deterministic relationship between physical reality and sensor output.

In this report we study some properties of particle filter as a statistically consistent tool for multiple frame low-resolution video information fusion. We apply it to vehicle movement model parameter estimation. Some experimental results presented in the report show that particle filter is able to track and detect moving objects even if very loose assumptions are made regarding the appearance of the objects and the values of motion model parameters. These results indicate that particle filter is able to learn movement model of a vehicle object even if availability of good

prior information regarding the values of these parameters is not assumed. Thus the estimated parameters of the motion model such as velocity can be used for traffic analysis.

A. Related work

Santini [1] suggests that most of the information regarding traffic behaviour is enclosed in the differences between adjacent frames. He notes also that changes in camera viewpoint and lighting conditions have rather profound effect on image differences. However, he concludes that changes due to traffic i. e. due to cars appearing and disappearing from images have local nature, whereas changes in lighting conditions and camera position cause global change in variance of inter-frame differences. Santini proposes to use 2-D inter-frame sample variance as a sufficient statistic for traffic flow estimation system. To assess the global status of the traffic using the measurements of traffic from separate cameras located at different crossroads he uses network tomography approach. The drawback of this approach is that it provides only qualitative characterization of the traffic flow.

An example of work dealing with data driven camera calibration and velocity estimation is given in [2]. What makes this work interesting is that instead of estimating motion vectors necessary for traffic velocity estimation from coded images of possibly low quality Mbonye and Ferrie [2] use partial decoding of MPEG stream to use motion vectors readily available within this stream. This procedure relies on an existing algorithm by Coimbra and Davies [3] that is able to extract one motion vector for each 16×16 MPEG macroblock. The application of this algorithm results in fast extraction of smoothed motion field. Kalman filter is used to track the velocity field estimate from frame to frame. However, for this method to work properly some assumptions regarding the road structure should hold. For example, there should be two lanes with the opposite directions of traffic flow. In some cases such assumptions may be too restrictive.

The algorithm proposed by Dailey et al. [4] belongs to the class of geometric algorithms relying on the availability of high frame rate video streams. Its main value is that before this algorithm was introduced little work had been done on velocity estimation using un-calibrated cameras. Camera calibration is very hard to handle in practice as was pointed out in [4]. This is especially

true when dealing with highway cameras that change their viewpoints several times every day to allow operator to better observe traffic conditions. This algorithm overcomes the difficulty by inferring calibration parameters directly from traffic images using relationship between the distribution of car sizes in pixels extracted from image sequence and some a priory known distribution of car sizes in meters. It uses three successive frames to form the differences between adjacent frames and extract edges using Sobel edge detector. To estimate the speed of moving cars, relationships between bounding boxes in three subsequent frames are established. This means that overall this algorithm requires the moving object to be present in five consecutive frames. The weak spots of the algorithm are as follows. First, it is necessary to make direct correspondence between object features appearing in three subsequent frames. These features may be very unstable because of the non-rigidity of the motion, camera jitter and noise, especially when low-resolution video is used for feature extraction. Second, some universal car length distribution that is applicable to a wide variety of traffic conditions should be specified.

Condensation algorithm was introduced in [5]. It belongs to the class of particle filtering algorithms that are capable of approximately solving general Bayesian state estimation problem. This algorithm is capable of tracking moving objects in image sequences using static feature extraction. However, it relies on availability of good initialization for object locations and parameters of movement model. This algorithm by itself is not suitable for movement model parameter estimation.

B. Outline of the report

This report is organized as follows. Section II reviews the general nonlinear state-space model that is adopted in Bayesian state estimation framework, particle filtering as an efficient Monte Carlo approximation of Bayesian solution to state estimation, and discusses feature extraction mechanism, dynamical and observation models used in Condensation algorithm. Section III provides a concrete statement of the particular problem and motivation behind this study. Section IV outlines modifications to the standard particle filter that make it possible to use this algorithm as a vehicle velocity estimator. Section V describes simulation experiments and discusses the results. Section VI provides concluding remarks.

II. Background

A. Nonlinear state-space model

Many problems in object tracking may be solved by the application of a state-space estimation framework [5]. The state-space estimation approach is based on the following signal model:

$$\mathbf{x}_t = f_x(\mathbf{x}_{t-1}) + \mathbf{v}_t \tag{1}$$

$$\mathbf{y}_t = f_y(\mathbf{x}_t) + \mathbf{u}_t \tag{2}$$

where $t = 1, 2, \ldots$ is the discrete time, $\mathbf{x}_t$ denotes the state vector, and $\mathbf{y}_t$ indicates the measurement obtained at time t. The state of the system evolves according to the stochastic difference equation (1) characterized by the nonlinear mapping $f_x : \mathbb{R}^x \rightarrow \mathbb{R}^x$ and excited by white random noise $\mathbf{v}_t$. The state vector $\mathbf{x}_t$ is observed through the measurement vector $\mathbf{y}_t$, which is functionally related to $\mathbf{x}_t$ via the function $f_y : \mathbb{R}^x \rightarrow \mathbb{R}^y$ and corrupted by white random noise $\mathbf{u}_t$. This framework can be applied to Computer Vision object tracking problems if we assume that equation (1) describes dynamics of some object in the sequence of images and equation (2) provides the link between the unobserved state $\mathbf{x}_t$ and feature vector $\mathbf{y}_t$ we extract from the image sequence.

B. Particle filtering

As discussed in [5], a practical Bayesian approach to the estimation of unobservable state $\mathbf{x}_t$ from the collection of measurements $\mathbf{y}_{1:t} \triangleq \{\mathbf{y}_1, \ldots, \mathbf{y}_t\}$ available up to time t consists in sequential construction of prediction and update densities. This approach is very similar to Kalman filtering concept except for the fact that very loose assumptions regarding nature of noise and transition functions are made.

To derive recursive Bayesian filtering equations we need to suppose that posterior distribution from the previous filtering step $p(\mathbf{x}_{t-1}|\mathbf{y}_{1:t-1})$ is available and we can use it to construct a prior density for current filtering step by prediction:

$$p(\mathbf{x}_t|\mathbf{y}_{1:t-1}) = \int p(\mathbf{x}_t|\mathbf{x}_{t-1}) p(\mathbf{x}_{t-1}|\mathbf{y}_{1:t-1}) \mathrm{d}\mathbf{x}_{t-1} \tag{3}$$

It should be noted, however, that strictly speaking $p(\mathbf{x}_t|\mathbf{y}_{1:t-1})$ is not a prior density at time step t. It is more reasonable to consider this density as a prediction of state distribution from posterior density using state dynamics $p(\mathbf{x}_t|\mathbf{x}_{t-1})$ arising from state transition equation (1)

$$p(\mathbf{x}_t|\mathbf{x}_{t-1}) = \int p(\mathbf{x}_t|\mathbf{x}_{t-1}, \mathbf{v}_{t-1})p(\mathbf{v}_{t-1}|\mathbf{x}_{t-1})\mathrm{d}\mathbf{v}_{t-1} \tag{4}$$

Using Bayes rule and Markov assumption embodied in equation (1) updated posterior density of state can be obtained after measurement $\mathbf{y}_t$ has arrived

$$p(\mathbf{x}_t|\mathbf{y}_{1:t}) = \frac{p(\mathbf{y}_t|\mathbf{x}_t)p(\mathbf{x}_t|\mathbf{y}_{1:t-1})}{p(\mathbf{y}_t|\mathbf{y}_{1:t-1})}. \tag{5}$$

Here normalizing constant $p(\mathbf{y}_t|\mathbf{y}_{1:t-1})$ has the following form:

$$p(\mathbf{y}_t|\mathbf{y}_{1:t-1}) = \int p(\mathbf{y}_t|\mathbf{x}_t)p(\mathbf{x}_t|\mathbf{y}_{1:t-1})\mathrm{d}\mathbf{x}_t \tag{6}$$

Likelihood $p(\mathbf{y}_t|\mathbf{x}_t)$ of state $\mathbf{x}_t$ can be evaluated using known measurement noise distribution:

$$p(\mathbf{y}_t|\mathbf{x}_t) = \int p(\mathbf{y}_t|\mathbf{x}_t, \mathbf{u}_t)p(\mathbf{u}_t)\mathrm{d}\mathbf{u}_t \tag{7}$$

Equations (3) and (5) constitute the basis of recursive Bayesian state estimation framework. However, closed form solutions to these recursions can be found for a limited class of state-space models including linear Gaussian models and finite state-space representations of the Markovian model (1). In all other cases approximate numerical methods must be used.

Monte Carlo *particle filters* belong to the class of sequential approximation algorithms capable of solving (approximately) the problem (3), (5) by direct numerical simulation [6]. In this class of algorithms, the filtering distribution (5) is represented by the empirical point mass function of the particle set

$$p(\mathbf{x}_t|\mathbf{y}_{1:t}) \approx \sum_{i=1}^{N} w_t^i \delta(\mathbf{x}_t - \mathbf{x}_t^i) \quad , \quad \sum_{i=1}^{N} w_t^i = 1 \tag{8}$$

where $\delta(\cdot)$ is the Dirac delta function and w_t^i denotes the normalized weight of each particle $\mathbf{x}_t^i$. The particle filter is initialized with a random sample $\{\mathbf{x}_0^i\}_{i=1}^N$ drawn from a known prior

distribution $p(\mathbf{x}_0)$. Subsequent propagation of this particle set using the equations (3), (5) yields at every time step the approximation of state vector by a discrete statistical measure of the form (8). Given the approximation of posterior distribution in the form of pmf the estimate of any moment of the posterior can be expressed in the following form:

$$\widehat{E}\{g(\mathbf{x}_t)|\mathbf{y}_{1:t}\} = \sum_{i=1}^{N} w_t^i g(\mathbf{x}_t^i) \tag{9}$$

The assumption in (8) is that it is possible to draw samples directly from posterior distribution $p(\mathbf{x}_t|\mathbf{y}_{1:t})$. However, in most practical situations it is impossible. That is why in practice particles are drawn from some easy to sample distribution $q(\mathbf{x}_t|\mathbf{y}_{1:t})$, which is called a proposal distribution. The required property of proposal distribution is that its support should be at least equal to the support of the true posterior distribution. It is possible to include proposal distribution into particle propagation steps (3) and (5) and derive a rule for sequential particle weight update based on Markovian assumption and importance sampling:

$$\tilde{w_t} = w_{t-1}\frac{p(\mathbf{y}_t|\mathbf{x}_t)p(\mathbf{x}_t|\mathbf{x}_{t-1})}{q(\mathbf{x}_t|\mathbf{x}_{0:t}, \mathbf{y}_{1:t})} \tag{10}$$

This technique for particle weight update is called Sequential Importance Sampling (SIS). It is shown for example in [6] that the unconditional variance of particle weights grows with time when SIS is used to update weights. This in turn leads to the increase of the variance of state estimate. The solution to this problem was found in the form of Sampling Importance Resampling (SIR) particle filter. In SIR particle filter a resampling step is introduced to substitute particles with low importance weights w_t^i by the particles with high importance weights. Resampling transforms current pmf approximation $\{\mathbf{x}_t^i, w_t^i\}_{i=1}^N$ with normalized weights $\{w_t^i\}_{i=1}^N$ to the equally weighted pmf $\{\mathbf{x}_t^{*i}, \frac{1}{N}\}_{i=1}^N$. Here the probability of each particle representation of posterior state pdf before resampling step $\{\mathbf{x}_t^i\}_{i=1}^N$ being resampled and included into particle representation of posterior state pdf after resampling step $\{\mathbf{x}_t^{*i}\}_{i=1}^N$ is equal to the value of corresponding normalized weight $\{w_t^i\}_{i=1}^N$.

C. Particle filtering in image processing: Condensation

1) Feature extraction: Feature extraction is an indispensable part of the Condensation algorithm. At every time step features are extracted from a still image. The sequence of such features is then tracked using observation and dynamics densities introduced in section II-B. The authors [5] propose to use a B-spline model to parameterize tracked objects. Within this framework objects are modeled as curves that can be parameterized in terms of B-splines in the following manner:

$$\mathbf{r}(s,t) = (\mathbf{B}(s) \cdot \mathbf{Q}^x(t), \mathbf{B}(s) \cdot \mathbf{Q}^y(t)) \qquad \text{for} \quad 0 \leq s \leq L \tag{11}$$

where $\mathbf{B}(s) = (B_1(s), B_2(s), \ldots, B_{N_B}(s))^T$ is a B-spline basis vector, $\mathbf{Q}^x$ and $\mathbf{Q}^y$ are B-spline control point vectors, and L is the number of spans. Parametrization in the form (11) allows representation of an object as a deformation of some template curve.

2) Dynamical Model: Object dynamics in Condensation algorithm are modeled according to a linear model corresponding to the following 2nd order difference equation:

$$\mathbf{x}_t - \bar{\mathbf{x}} = \mathbf{A}(\mathbf{x}_{t-1} - \bar{\mathbf{x}}) + \mathbf{B}\mathbf{v}_t \quad , \quad \mathbf{x}_t = \begin{pmatrix} \mathbf{X}_{t-1} \\ \mathbf{X}_t \end{pmatrix} \tag{12}$$

where $\bar{\mathbf{x}}$ is the mean value of the state vector, $\mathbf{A}$ is the deterministic transition matrix, and $\mathbf{B}$ is the matrix defining the way in which random excitation $\mathbf{v}_t$ interacts with the deterministic part of the dynamic system. It can be seen that model (12) is a special case of non-linear model (1). Linear model was chosen by the authors because it is simple and a variety of techniques exists for learning it on-line.

3) Observation Model: The observation model that is used in Condensation algorithm includes a probabilistic mapping between point features in the image and B-spline curve parameters. In practice it is assumed that there exists some mapping $g(s)$ between measured features $\mathbf{y}(s)$ and normals to the curve $\mathbf{r}(g(s))$ such that a discrete approximation to the 2-D likelihood function $p(\mathbf{y}|\mathbf{x})$ can be written in the following form:

$$p(\mathbf{y}|\mathbf{x}) \propto \exp\left(-\frac{1}{2rM}\sum_{m=1}^{M} f(\mathbf{y}_1(s_m) - \mathbf{r}(s_m); \mu)\right) \tag{13}$$

Here $s_m = m/M$, $\mu = \sqrt{2}\sigma \log(1/\sqrt{2\pi}\alpha\sigma)$ is the spatial scale constant, and f is defined as follows $f(\nu;\mu) = \min(\nu^2, \mu^2)$. This parametrization of likelihood function is obtained by the authors using the assumptions that the clutter can be modeled as a Poisson process and measurement is an unbiased estimator of the true feature location having Gaussian pdf and variance $\sigma^2 = rM$. It should be noted that the measurement process parametrization used in the Condensation algorithm does not take into account the fact that tracked objects move from frame to frame. On the one hand, this static feature extraction approach is reasonable because it is not affected by the effects of camera movement when the whole scene may be moving. On the other hand, vision system of many animals that chase after moving prey does use movement information for feature extraction and object detection. For example, frogs are almost blind if objects in the scene do not move. Roughly speaking, frogs can see only those objects that move because of the peculiarities in their retina scanning process, to be precise, because of the absence of this scanning process. Nevertheless, and may be in many respects owing to this, frogs are excellent hunters. In this sense, static feature extraction can even be considered a counterintuitive measure to detect moving objects.

III. Problem statement and study motivation

In this report we would like to explore the potentials of a particle filter as a standalone velocity estimator, moving object tracker, and detector. An interesting application for such an algorithm would be vehicle traffic estimation in low resolution image sequences obtained from camera networks that exist along many highway lanes. Because of the low quality of image sequences that are often available for processing in practice, static feature extraction and object detection in such sequences may be difficult and unreliable. More than this, standard approaches to velocity estimation in traffic data often use geometrical velocity measurement based on calibrated cameras and Kalman filter relying on Gaussian measurement error to smooth measurements. In low resolution images camera calibration and geometrical velocity estimators may be very unreliable and measurement error may have very heavy tailed distribution. In this situation even the applicability of Kalman filter may become questionable. On the other hand, particle filter is a statistically consistent procedure for multiple frame information fusion that is

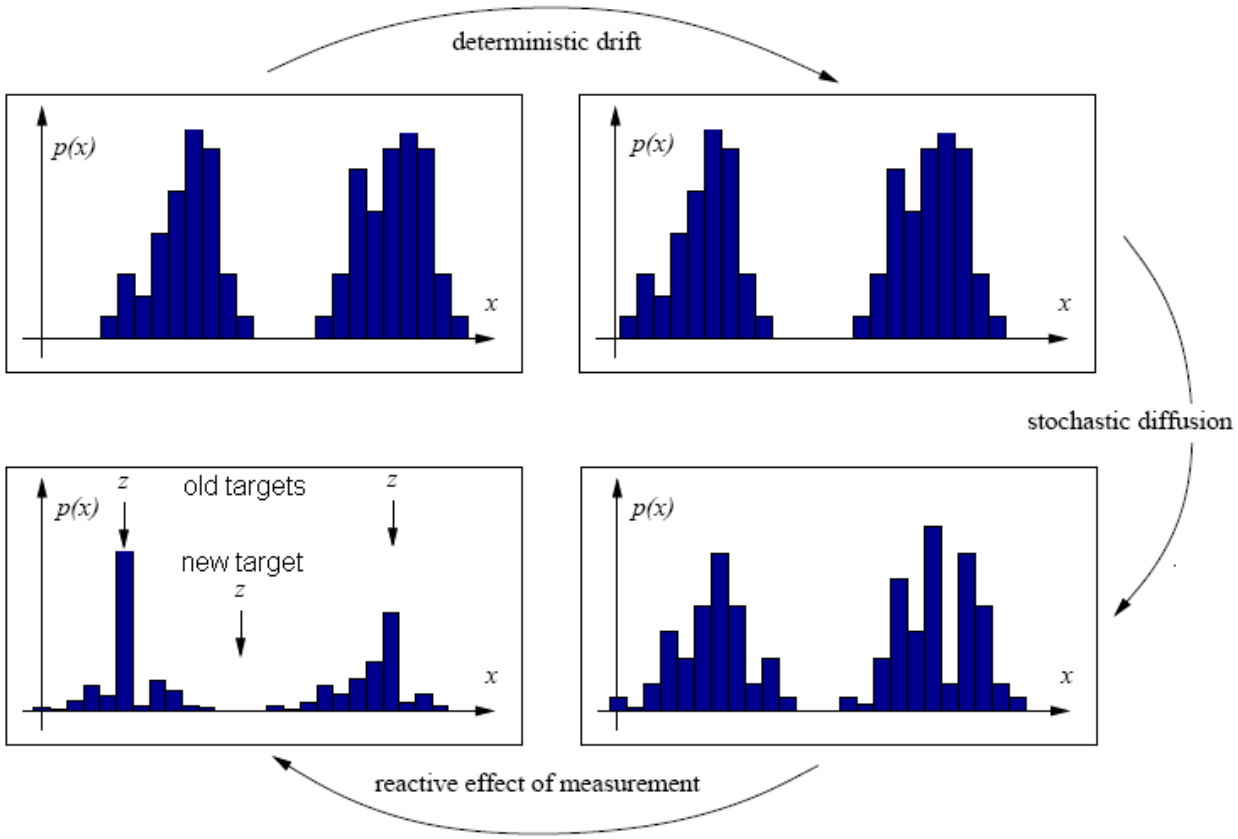


Fig. 1. Concentration of particles in the areas supporting already tracked objects prevents the detection of an object newly entering the scene. Taken from [5] in a modified form

necessary in situation when every frame or measurement by itself may bear little information. The core benefits of using particle filters for tracking and parameter estimation in low resolution video sequences are their ability to automatically recover after missing measurements and track loss, inherent ability to track multiple targets simultaneously within single filter, and applicability of the filter to any observation model.

Thus it is of interest to see how particle filter can handle parameter estimation and tracking of multiple moving objects in low resolution video scenario. To be able to detect and track moving objects we would like to use some inter-frame information encapsulating the assumption that objects are moving, i.e. change their position from frame to frame. In this case observation model might represent the likelihood of every pixel or a group of pixels being a part of a moving object footprint. If particle filter is able to generate the spikes of posterior density around moving objects in feature space then moving object locations as well as movement parameters can be extracted

directly from the particle approximation to posterior density. However, standard particle filtering algorithm that is widely known in image processing under the name Condensation assumes that a good initialization for object locations and movement model is available. Once it converges to tracking objects already present in the scene it is often not able to take into account objects newly entering into the scene. This happens because objects that are already tracked concentrate most of the particles around themselves thus making new object detection very improbable. This situation is illustrated in Fig. 1 showing the propagation of the discrete posterior density approximation resulting in the blanking of a new object.

In this report we present a particle filtering algorithm that overcomes the problem of unknown movement parameters and absence of good initialization for moving object locations by the application of a specific sampling strategy and movement model parameter estimation. As a byproduct of this modification we obtain and study the procedure for vehicle velocity estimation. This velocity estimator can then be used for traffic intensity assessment. In traffic intensity assessment we can use the ratio of instantaneous velocity to the maximum traffic velocity estimated during long periods of traffic analysis to characterize current traffic state. This kind of traffic state characterization does not require camera calibration that can be difficult or even impossible in many situations.

IV. Particle filter for velocity estimation

A. Dynamical model

We modify the linear dynamical model outlined in equation (12) in the following way. First of all, we do not assume that any prior information is available as to what the mean state vector $\bar{\mathbf{x}}$ might be. Thus our model takes the following form:

$$\mathbf{x}_t = \mathbf{F}\mathbf{x}_{t-1} + \mathbf{G}\boldsymbol{\xi}_t \tag{14}$$

Second, as we are interested in velocity estimation, the target state vector is appended with the vector of unknown velocities. This technique is often used in joint recursive state and parameter

estimation [7]. Thus the exact parametrization of (14) is as follows:

$$\mathbf{x}_t = \begin{pmatrix} x_{1,t} \\ x_{2,t} \\ v_{1,t} \\ v_{2,t} \end{pmatrix} \quad , \quad \mathbf{F} = \begin{pmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{15}$$

$$\boldsymbol{\xi}_t = \begin{pmatrix} \xi_{1,t} \\ \xi_{2,t} \end{pmatrix} \quad , \quad \mathbf{G} = \begin{pmatrix} \frac{1}{2} & 0 \\ 0 & \frac{1}{2} \\ 1 & 0 \\ 0 & 1 \end{pmatrix} \tag{16}$$

Here $x_{1,t}$ and $x_{2,t}$ are x and y coordinates of the target correspondingly, $v_{1,t}$ and $v_{2,t}$ are x and y velocity components, and $\xi_{1,t}$ and $\xi_{2,t}$ are x and y acceleration components.

Each particle i that is used in tracking $i = 1, \ldots, N_T$ is then assigned a state vector $\mathbf{x}_t^{(i)}$ corresponding to some pixel neighborhood $\mathcal{N}_i$ and tracking is performed on a neighborhood by neighborhood basis. We assume that objects perform rigid motion in feature space. Although in general this is not true in practice, our experiments show that particle filter is still able to capture average pixel dynamics during the tracking of non-rigidly moving object footprints in feature space. Which means that even if the object's appearance in feature space changes randomly from frame to frame, particle filter is still able to estimate velocity of this object with reasonable estimation error.

B. Image preprocessing

As was mentioned earlier, in our work we do not rely on a static feature extraction mechanism. Instead, we try to use the inter-frame information to estimate the likelihood of each pixel being a part of moving object footprint. As the direct identification and tracking of objects in low quality images is often impossible we first transform the sequence of the images into the feature space. The result of feature extraction should show how probable it is that pixel with coordinates (x_1, x_2) pertains to some moving object with state vector $\mathbf{x}_t^o = [x_1^o, x_2^o, v_1^o, v_2^o]^T$. Here v_1^o, v_2^o are x and y components of its velocity vector. We assume that the intensity of each pixel

$y(x_1, x_2)$ can be modeled as a Gaussian random variable with some unknown mean μ_p and variance σ_p^2: $y(x_1, x_2) \sim \mathcal{N}(\mu_p, \sigma_p^2)$. We assume also that if pixel with coordinates (x_1, x_2) in a sequence of frames pertains to a still object then the mean μ_p for this pixel in these frames is the same. If, however, this pixel pertains to the footprint of a moving object, the mean of this pixel corresponding to the frame at the time instant $t-1$ is μ_p^{t-1} and μ_p^t at the time instant t. Therefore, the best feature extraction procedure would implement an optimum statistical test to differentiate between the two hypotheses

$$H_0 : \mu_p^t = \mu_p^{t-1} \tag{17}$$

$$H_1 : \mu_p^t \neq \mu_p^{t-1} \tag{18}$$

given the measurements

$$y_t(x_1, x_2) = \mu_p^t(x_1, x_2) + \zeta(x_1, x_2) \tag{19}$$

where $\zeta(x_1, x_2) \sim \mathcal{N}(0, \sigma_p^2)$ is Gaussian random variable with zero mean.

However, it is known that the best optimum (uniformly most powerful) statistical test to differentiate between the two hypotheses (17) when μ_p^{t-1} and μ_p^t are arbitrary and unknown does not exist [8]. The following suboptimum test statistic and decision rule that are equivalent to the frame differencing and thresholding are used for feature extraction:

$$\Lambda(y_t(x_1, x_2), y_{t-1}(x_1, x_2)|x_1, x_2) \quad \propto \quad 1 - \exp\left(-\frac{(y_t(x_1, x_2) - y_{t-1}(x_1, x_2))^2}{2 \cdot 2\sigma_p^2}\right) \tag{20}$$

$$\Lambda(y_t(x_1, x_2), y_{t-1}(x_1, x_2)|x_1, x_2) \quad \underset{H_0}{\overset{H_1}{\gtrless}} \quad \eta_1 \tag{21}$$

C. Observation model

After the feature extraction, what we observe is the binary image that is formed as a result of the test (21) performed on successive frames:

$$z_t(x_1, x_2) = \begin{cases} 1 & \text{if} \quad \Lambda(y_t(x_1, x_2), y_{t-1}(x_1, x_2)|x_1, x_2) > \eta_1 \\ -1 & \text{if} \quad \Lambda(y_t(x_1, x_2), y_{t-1}(x_1, x_2)|x_1, x_2) < \eta_1 \end{cases} \tag{22}$$

Simply following the observation model (13) used in Condensation algorithm we can reformulate it to fit our concrete case as follows:

$$p(z(x_1,x_2)|s(x_1,x_2),x_1,x_2) \propto \exp\left(-\frac{1}{2\Delta_l M_{\mathcal{N}_i}} \sum_{x_1,x_2 \in \mathcal{N}_i} (z(x_1,x_2)-s(x_1,x_2))^2\right) \tag{23}$$

Where $M_{\mathcal{N}_i}$ is the number of pixels in the neighbourhood $\mathcal{N}_i$, Δ_l is the width of the likelihood kernel and $s(x_1,x_2), x_1,x_2 \in \mathcal{N}_i$ is the true appearance (footprint) of the tracked object in feature space. It is clear that (24) can be reformulated in a more concise way:

$$p(z(x_1,x_2)|s(x_1,x_2),x_1,x_2) \propto \exp\left(-\frac{1}{\Delta_l}(1-\rho)\right) \tag{24}$$

where

$$\rho = \frac{1}{M_{\mathcal{N}_i}} \sum_{x_1,x_2 \in \mathcal{N}_i} s(x_1,x_2) z(x_1,x_2) \tag{25}$$

is the generalization of the so called Hough Transform (HT). When $s(x_1,x_2) \equiv 1 \ \ \forall x_1,x_2 \in \mathcal{N}_i$, (25) exactly represents the idea of HT. Also, (25) can be considered as an expression for a correlation coefficient between $s(x_1,x_2)$ and $z(x_1,x_2)$.

It should be noted that the conditional distribution $p(z(x_1,x_2)|s(x_1,x_2),x_1,x_2)$ is not at all Gaussian and (24) is at best an approximation of the real observation density. However, our experiments show that even with this rough approximation of observation density particle filter works reasonably well. More than this, $p(z(x_1,x_2)|s(x_1,x_2),x_1,x_2)$ can be calculated exactly, given all the previous assumptions and comparison of the filter with optimum observation density and approximation (24) would be an interesting thing to do.

D. Sampling scheme

Often in practical particle filters and also in Condensation algorithm, transition density $p(\mathbf{x}_t|\mathbf{x}_{t-1})$ is picked as a proposal density $q(\mathbf{x}_t|\mathbf{y}_{1:t})$. In our experiments we use a slightly modified sampling scheme that is depicted in Fig. 2. The difference here is that two distinct particle clouds are used to track and detect moving objects. As was mentioned earlier, existing moving targets attract most of the particles to their locations preventing the exploration of other spatial areas of the image sequence. To overcome this difficulty in the setting where no prior information is

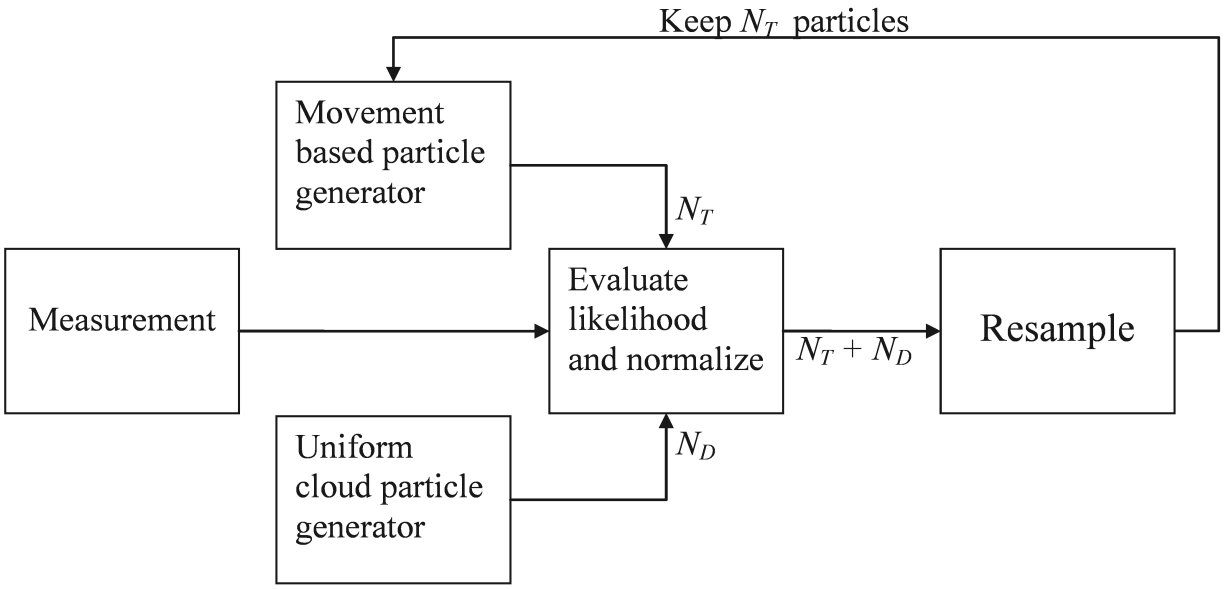


Fig. 2. Sampling scheme that is used in particle filter with simultaneous detection and tracking of moving targets

available regarding the location of every new object appearing in the tracked scene we introduced a separate search cloud uniformly distributed over the entire image area. This cloud supplements tracking cloud that contains the information about those objects that are already tracked. Both clouds undergo particle weight evaluation using observation mechanism outlined in section IV-C. In general they contain different number of particles. As was mentioned earlier, the track cloud contains N_T particles. At the same time, search cloud contains N_D particles. After weight evaluation and before weight normalization weight vectors of both clouds are appended. Thus during the resampling process particles with significant weights originating from the search cloud substitute those with low weights originating from the track cloud. Generally speaking, the mixture of these two clouds can be implemented by the appropriate choice of the proposal distribution. However, it might be more intuitive to imagine this mixture as two dedicated clouds.

E. Velocity estimation

We have already mentioned earlier that velocity of the moving object can be estimated by the particle filter if this velocity is included into the state vector as an unknown variable [7]. This technique is sometimes called sample roughening. It was recognized in [7] that the application of sample roughening to static parameters of the model may (and often does) cause inadequate

diffusion of posterior particle distribution leading to the large variance of velocity estimator and even filter divergence. Therefore, it might be beneficial to try and treat velocity as an unknown parameter $\theta(x_1, x_2)$ in transition density. Where $\theta(x_1, x_2)$ is a matrix, that is, it is defined as a field over the entire image area. We further assume that at any point (x_1, x_2) $\theta(x_1, x_2)$ assumes some arbitrary value independent of all the others. A valid way to estimate this parameter then is to find the argument maximizing evidence [9]:

$$p_\theta(\mathbf{y}_t|\mathbf{y}_{1:t-1}) = \iint p(\mathbf{y}_t|\mathbf{x}_t)p_\theta(\mathbf{x}_t|\mathbf{x}_{t-1})p_\theta(\mathbf{x}_{t-1}|\mathbf{y}_{1:t-1})d\mathbf{x}_{t-1:t} \tag{26}$$

One way to deal with such recursive maximization is to consider the Monte Carlo approximation of (26) that is a simple sum of unnormalized importance weights:

$$\widehat{p_\theta}(\mathbf{y}_t|\mathbf{y}_{1:t-1}) = \sum_{i=1}^{N_T} w_t^{(i)} \tag{27}$$

Using the properties of Monte Carlo density approximation we can show that as $N_T \to \infty$, the maximization of (27) is almost surely equivalent to the maximization of (26). Thus we can use the following problem statement to find a recursive estimator of θ:

$$\theta_t = \arg\max_{\theta_t} \sum_{i=1}^{N_T} w_t^{(i)} \tag{28}$$

Now, if we recall the way in which θ comes into the play in $p_\theta(\mathbf{x}_t|\mathbf{x}_{t-1})$ (14) and apply the velocity field independence assumption, then instead of (28) we can solve a simpler problem:

$$\theta_t(x_1, x_2) = \arg\max_{\theta_t(x_1,x_2)} w_t^{(i)}(x_1, x_2) \tag{29}$$

Where x_1, x_2 are x, y components of particle state vector. To solve (29) using Monte Carlo technique, we can approximate the filtering density $p_\theta(\mathbf{x}_{t-1}|\mathbf{y}_{1:t-1})$ by N_T particles and during the update step we can expand this particle cloud by generating N_v diffused particles from every particle of posterior density. After that, we can easily solve the problem (29) by just selecting the particle with the largest weight and discarding all the rest $N_v - 1$ particles.

Using the steps outlined above we generate the particle approximation of the velocity estimator. In order to estimate velocity at any desired point x_d, y_d of the image we apply kernel smoothing

step [7] as described below. Normalized importance weights of the track cloud:

$$\widetilde{w}^i_{(T)t} = \frac{w^i_{(T)t}}{\sum_{j=1}^{N_T} w^j_{(T)t}} \tag{30}$$

and kernel weights reflecting the closeness of every particle to the desired location:

$$w^i_{(k)t} = \exp\left(-\frac{(x^{(i)}_{1,t} - x_d)^2 + (x^{(i)}_{2,t} - y_d)^2}{4\Delta_k}\right) \tag{31}$$

where Δ_k is kernel width equal to 6 pixels are multiplied and velocity estimation weights are obtained:

$$w^i_{(v)t} = \widetilde{w}^i_{(T)t} w^i_{(k)t} \tag{32}$$

Smoothed velocity estimator at desired position $v^d_{j,t}$ is then formed as a weighted sum of particle velocities:

$$v^d_{j,t} = \frac{1}{\sum_{p=1}^{N_T} w^p_{(v)t}} \sum_{i=1}^{N_T} w^i_{(v)t} v^{(i)}_{j,t} \tag{33}$$

F. Algorithm

In this section we summarize the algorithm of particle filtering for velocity estimation. This algorithm is presented in Fig. 3

For t = 0,1,2,...

1) Get image: $\mathbf{y}_t$
2) Image preprocessing
   - Calculate decision statistic (frame differencing):
     $\Lambda(y_t(x_1,x_2), y_{t-1}(x_1,x_2)|x_1,x_2) = 1 - \exp\left(-\frac{(y_t(x_1,x_2)-y_{t-1}(x_1,x_2))^2}{2\cdot 2\sigma_p^2}\right)$
   - Calculate feature space (thresholding):
     $$z_t(x_1,x_2) = \begin{cases} 1 & \text{if} \quad \Lambda(y_t(x_1,x_2), y_{t-1}(x_1,x_2)|x_1,x_2) > \eta_1 \\ -1 & \text{if} \quad \Lambda(y_t(x_1,x_2), y_{t-1}(x_1,x_2)|x_1,x_2) < \eta_1 \end{cases}$$
3) Particle filtering
   - Generate search cloud: For $i = 1$ to N_D, $\mathbf{x}_t^i \sim [\mathrm{U}(0,w), \mathrm{U}(0,h), 0, 0]^T$;
   - Evaluate likelihood of the search cloud: For $i = 1$ to N_D,
     $\rho_{(D)t}^{(i)} = \frac{1}{M_{\mathcal{N}_i}} \sum_{x_1^{(i)}, x_2^{(i)} \in \mathcal{N}_i} s_t(x_1^{(i)}, x_2^{(i)}) z_t(x_1^{(i)}, x_2^{(i)})$;
     $w_{(D)}^{(i)} = \exp\left(-\frac{1}{\Delta_l}(1 - \rho_{(D)t}^{(i)})\right)$;
   - Propagate track cloud and estimate velocity:

     For $i = 1$ to N_T,

     For $j = 1$ to N_v,

     $\mathbf{x}_t^{(i,j)} = \mathbf{F}\mathbf{x}_{t-1}^{(i)} + \mathbf{G}\boldsymbol{\xi}_t^{(i)}$;
     $\mathbf{x}_t^{(i,j)+} = \mathbf{F}\mathbf{x}_t^{(i)}$;

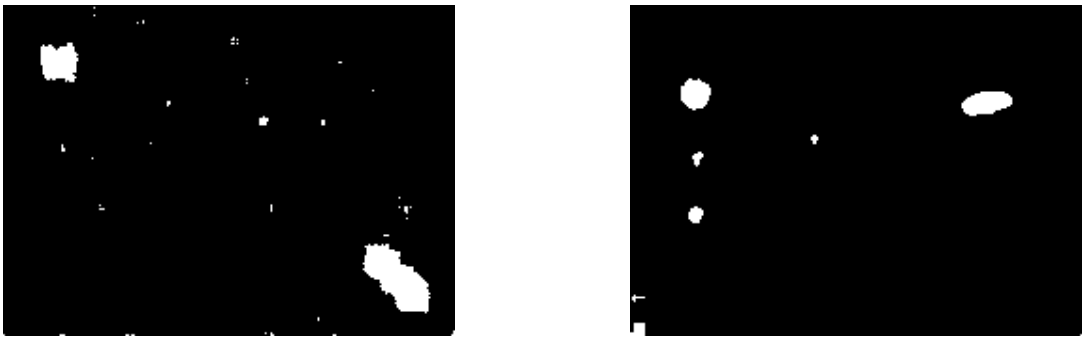
     – Evaluate likelihood of the track cloud:

     $\rho_{(T)t}^{(i,j)} = \frac{1}{M_{\mathcal{N}_i}} \sum_{x_1^{(i,j)}, x_2^{(i,j)} \in \mathcal{N}_i} s_t(x_1^{(i,j)}, x_2^{(i,j)}) z_t(x_1^{(i,j)}, x_2^{(i,j)})$;
     $\rho_{(T)t+1}^{(i,j)} = \frac{1}{M_{\mathcal{N}_i}} \sum_{x_1^{(i,j)+}, x_2^{(i,j)+} \in \mathcal{N}_i} z_t(x_1^{(i,j)}, x_2^{(i,j)}) z_{t+1}(x_1^{(i,j)+}, x_2^{(i,j)+})$;
     $w_{(T)}^{(i,j)} = \exp\left(-\frac{1}{\Delta_l}(1 - \rho_{(T)t}^{(i,j)})\right) \exp\left(-\frac{1}{\Delta_l}(1 - \rho_{(T)t+1}^{(i,j)})\right)$;

     $\mathbf{x}_t^{(i)} = \arg \max_{\mathbf{x}_t^{(i,1\ldots N_v)}} w_{(T)}^{(i,1\ldots N_v)}$;
   - Estimate velocity at particle locations using (30)–(33)
   - Append and normalize weight vectors: $w_t = [w_{(D)t}^T, w_{(T)t}^T]^T$;

     For $i = 1$ to $N_D + N_T$: $\widetilde{w}_t^i = \frac{w_t^i}{\sum_{j=1}^{N_D+N_T} w_t^j}$;
   - Resample

Fig. 3. Summary of particle filtering for velocity estimation



(a) Synthetic data, $\varrho = 0.2$ (b) Real data

Fig. 4. The example image of feature space generated using the simulator (left) and real camera data (right)

V. EXPERIMENTS

A. Synthetic model description

In our experiments we use the following synthetic model. The target is represented by a square set of pixels having size 15×15 pixels. Each time a target is generated, part of the pixels is marked as detected and part of them as not detected. During the movement of the target its pixels may switch from one state to the other with probability p_s according to a markov model. The movement of the target obeys linear constant velocity model. Noise in the image is generated as false alarms uniformly distributed over the entire image area with density ϱ. After this first stage image is generated, we apply image smoothing and after that second stage thresholding to get rid of the small artifacts and to close moving target contours. The example image of feature space generated using the simulator is shown in Fig. 4 (left). The image of feature space generated using real data is shown in Fig. 4 (right) for comparison. For a more comprehensive comparison we also supplement this report with a video example ”feat_sp_compare.avi”. This example contains the recording showing feature space generated by the simulator described above in the left pane and the feature space generated using real data by the image preprocessing step described in section IV-C.

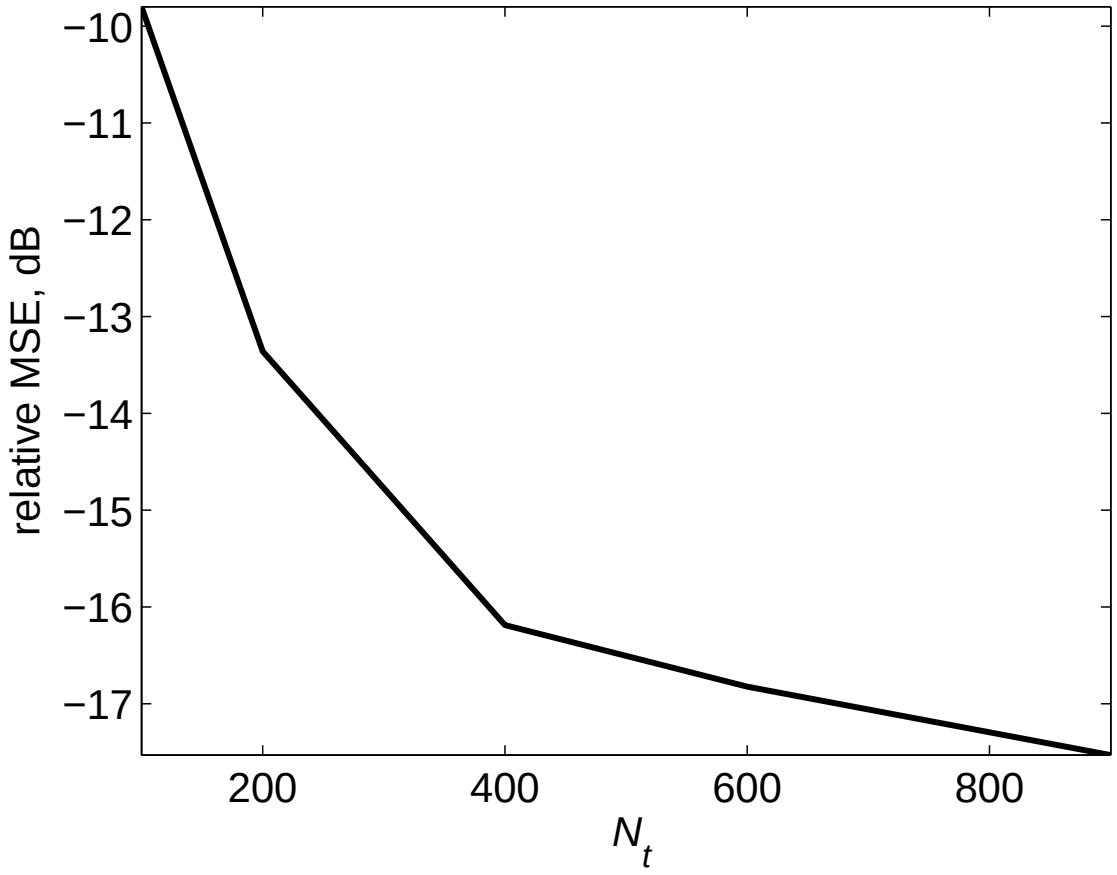


Fig. 5. Dependency of the MSE of velocity estimation from the number of particles in the search cloud N_T

B. Experiments with synthetic data

To assess the performance of the velocity estimation algorithm depicted in Fig. 3 a series of experiments on synthetic data was performed. The results of these experiments are presented in Fig. 5–9.

The dependency of the MSE of velocity estimation from the number of particles in the search cloud N_T is shown in Fig. 5. As expected, the estimation error reduces as the number of particles increases.

Next we present the plot showing the MSE of velocity estimation as a function of the number of particles N_v in velocity estimation step. During this experiment we keep the number of particles in the expanded cloud constant and equal to 4500. The number of particles N_v in velocity estimation step changes from 1 to 9 and the number of particles representing the posterior density changes

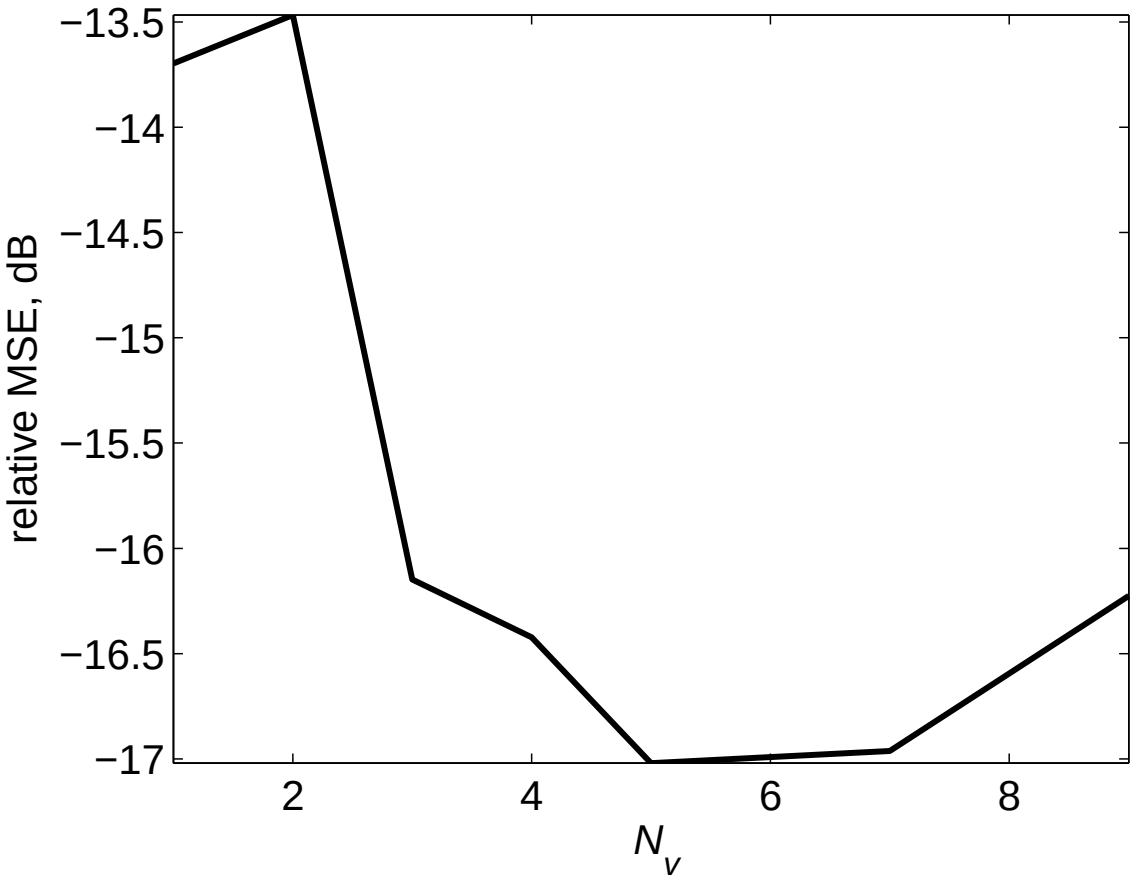


Fig. 6. MSE of velocity estimation as a function of the number of particles N_v in velocity estimation step

inversely to N_v, $N_T = 4500/N_v$. It is clear that in terms of computational complexity filters with different N_v are equivalent. It is interesting to note, that first the estimation error reduces as N_v grows, because this allows for better velocity estimation at every time step. But when N_v becomes greater than 5, MSE starts to increase. This can be attributed to the fact that further increase in the number of velocity estimation particles leads to the depletion of the number of particles that effectively represent the posterior filtering density, which in its turn leads to larger positioning (tracking) errors.

After that we study the influence of the absolute value of estimated velocity on the estimation error. The relative MSE decreases with the increasing absolute value of the velocity. Thus we can conclude that absolute MSE is approximately constant. This means that the described velocity estimation technique is suitable for the estimation of velocity in a practical range.

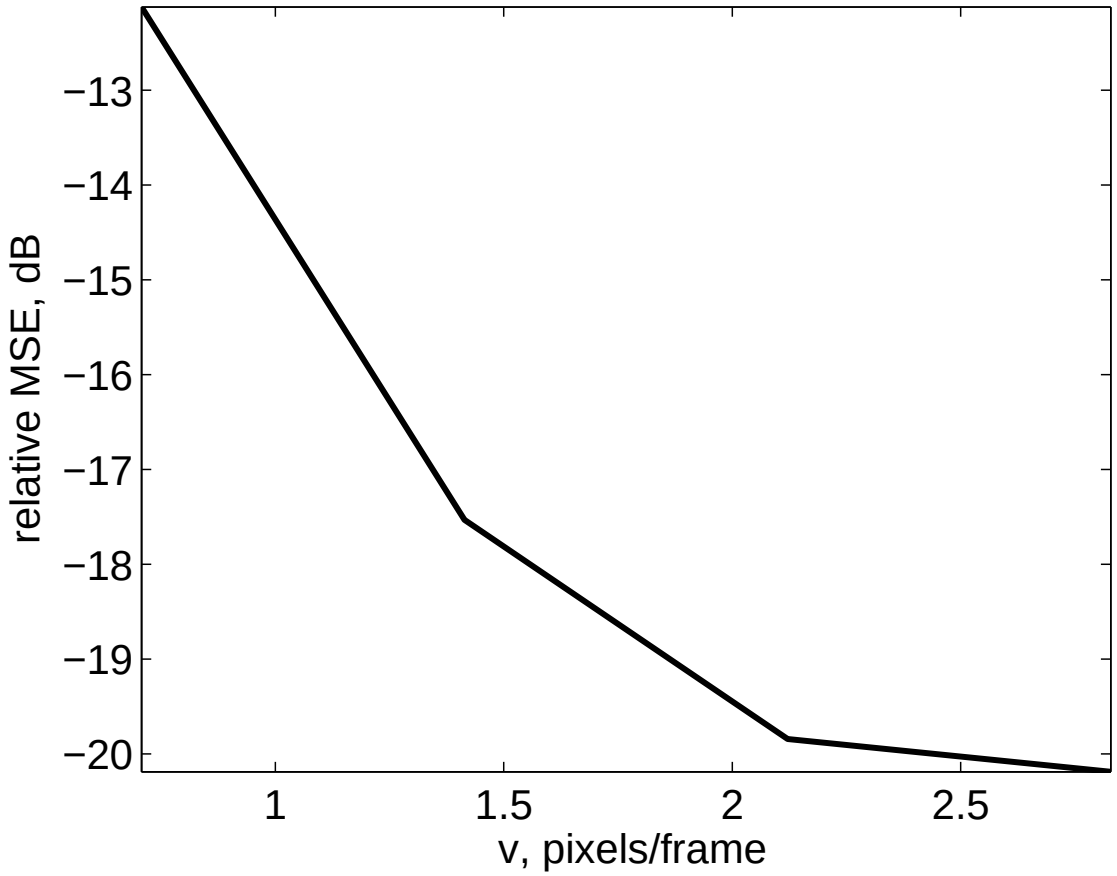


Fig. 7. MSE of velocity estimation as a function of the number of the absolute value of the velocity

Next we study the influence of the number of simultaneously tracked targets on the relative MSE. The relative MSE increases with the increasing number of tracked moving objects. This is an expected result, because each target requires a number of particles to be tracked. Thus each new target appearing in the scene reduces the number of particles per target and thus the error increases according to Fig. 5.

Finally, we present the plot of MSE versus the noise density. Noise density is the percentage of false alarms uniformly distributed over the image area in every frame. It can be seen in Fig. 9 that as noise density increases MSE of velocity estimation also increases. This can be explained by the following. First, the presence of noise makes object movements deformable. That is, the object's appearance in the next frame is always different from its appearance in the previous frame. Thus the target constantly looses some of the tracking particles because they

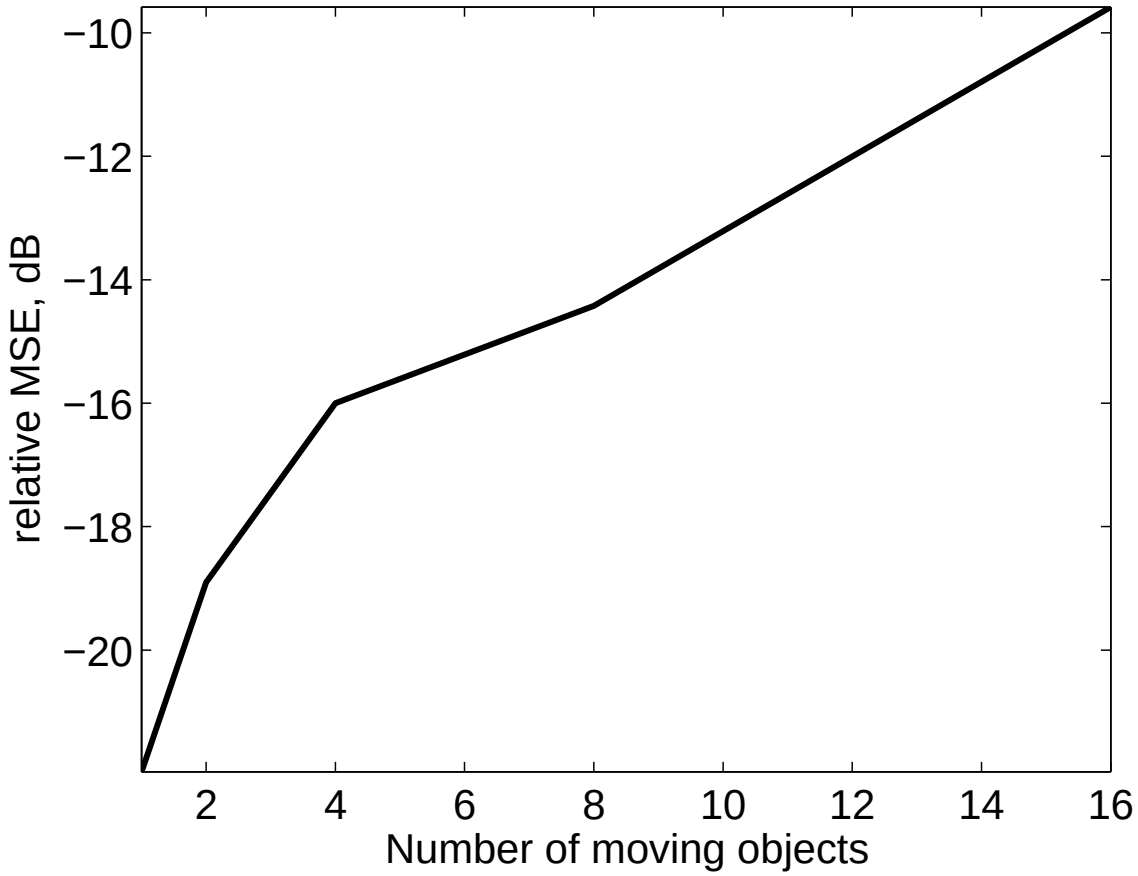


Fig. 8. MSE of velocity estimation as a function of the number of simultaneously tracked targets

assume low importance weights in the areas where deformable motion leads to the distortion of the true velocity profile. Second, when the noise level is relatively high, large noisy areas distract some particles from tracking cloud making the total number of particles useful for object tracking less. Also the nature of the error dependency from noise level suggests that tracking algorithm's performance deteriorates slowly as noise level increases and therefore suggested velocity estimation algorithm is robust to the presence of noise in data.

C. Experiments with real data

To experiment with the described particle filter for velocity estimation based on real data we used traffic monitoring data available on-line [10]. The concrete scene that was used during the simulation is shown in Fig. 10. In Fig. 11 we present the maximum absolute velocity profile

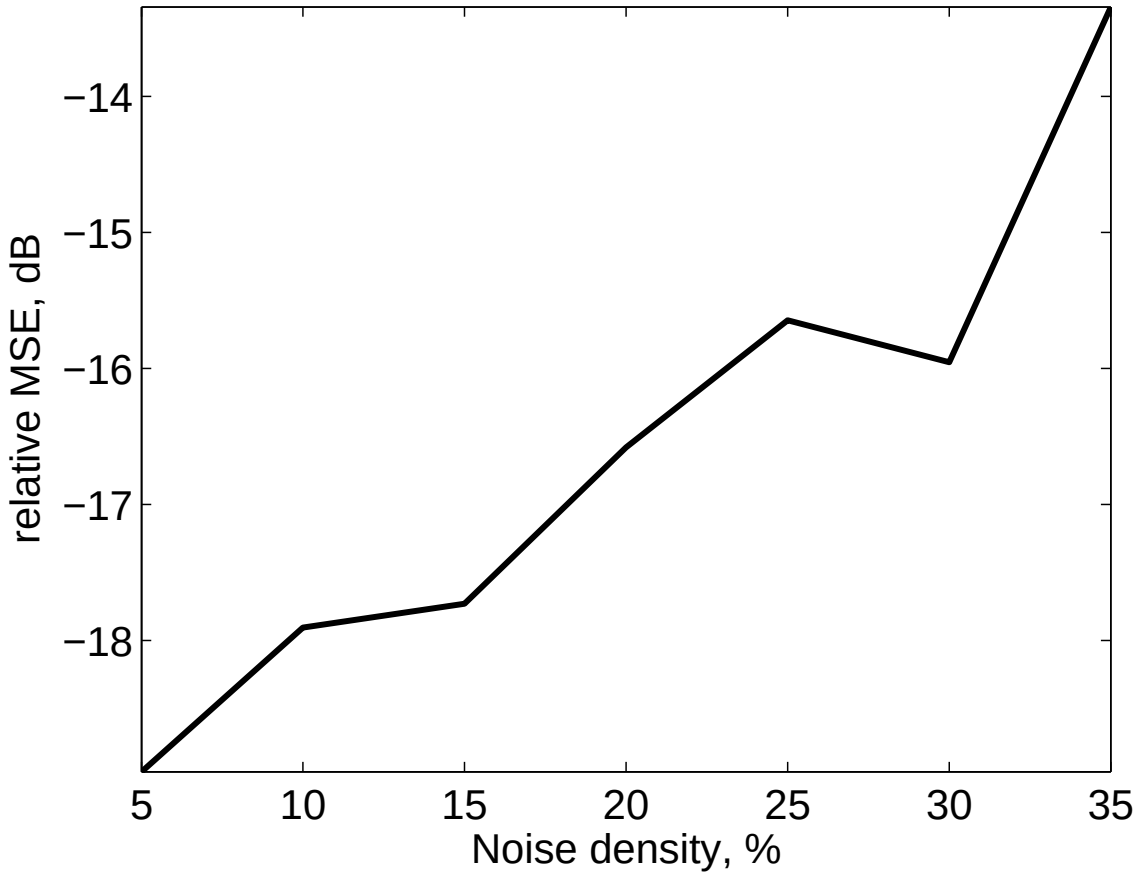


Fig. 9. MSE of velocity estimation as a function of the number of the absolute value of the velocity

learned by the particle filter while tracking the vehicles moving in this area. Also we supplement this report with the tracking video sequence ”track_results.avi”. Original image sequence is depicted in the higher left corner of this video. Original image sequence with the tracking results is shown in the higher right corner of this video. Here red dots depict particles from the search cloud and green dots depict particles from the track cloud. The appearance of the feature space corresponding to the video sequence is shown in lower right corner and the estimated velocity profile appears in the lower left corner. It can be seen from the tracking video sequence that although the amount of the prior information we use to construct the tracking model is small, particle filter is still able to autonomously detect moving objects marking them by red particles, assign green particles to track them and learn movement model as they move along the tracking area. All this confirms the validity of experimental results regarding velocity estimation

Fig. 10. The scene that was used during the simulation with real data

that were obtained using the simplified, but fully controlled synthetic model.

VI. Conclusion

In this report we experimented with particle filtering algorithm for velocity estimation in image sequences. In particular, we considered a situation where little a priory knowledge was used to parameterize exact appearance of the tracked objects and movement model was not initialized using good prior knowledge that is sometimes available from object detection step. However, particle filter was able to handle this situation and automatically allocate tracking resources to the moving objects newly entering the scene, estimate the velocity of moving objects on-line, and localize moving objects by creating posterior density spikes in the areas of the image corresponding to the objects of interest. We obtained useful experimental results using both synthetic and real data. These results can be used to assess the applicability of the

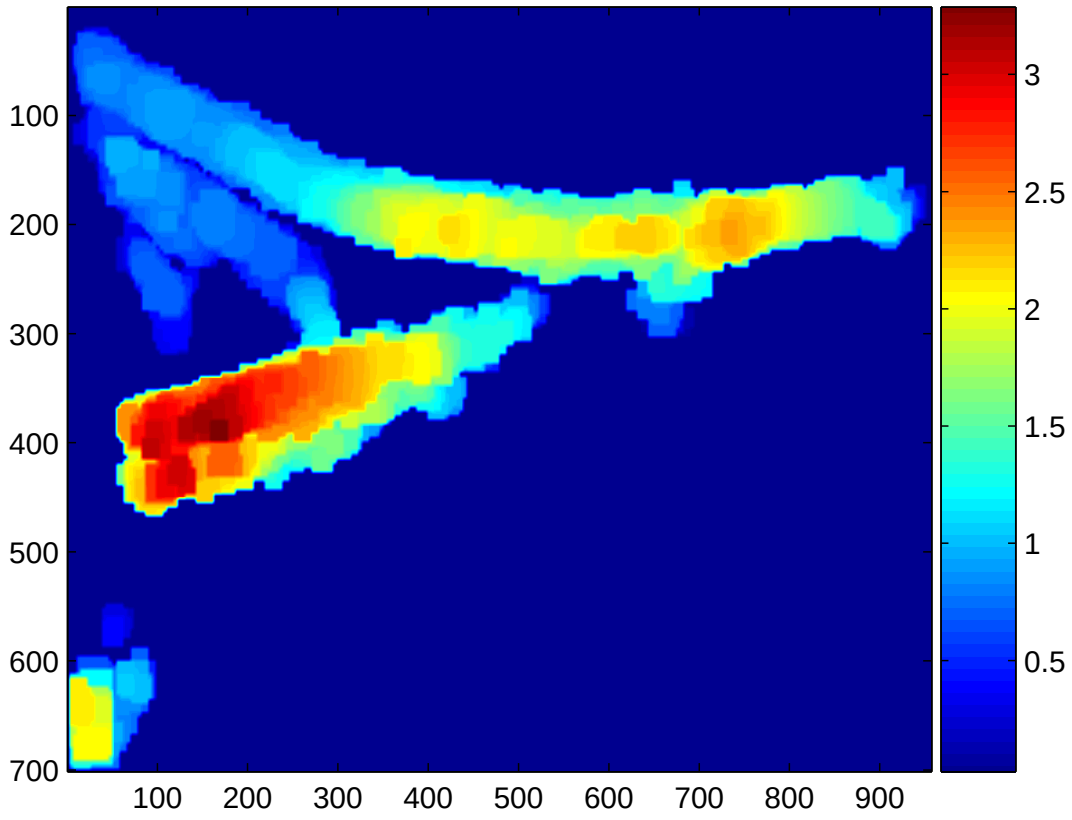


Fig. 11. Maximum absolute velocity profile learned by the particle filter while tracking the vehicles moving in the area shown in Fig. 10

studied algorithm to some particular situation and optimize the performance of the algorithm by choosing the optimum values of its parameters.

REFERENCES

[1] S. Santini, "Analysis of traffic flow in urban areas using web cameras," in *Proc. 5th IEEE Workshop on Applications of Computer Vision*, Palm Springs, CA, Dec. 2000, pp. 140–145.

[2] K. P. Mbonye and F. P. Ferrie, "Attentive visual servoing in the MPEG compressed domain for un-calibrated motion parameter estimation of road traffic," in *Proc. ICRP'06*, Washington, DC, Aug. 2006, pp. 908–911.

[3] M. T. Coimbra and M. Davies, "Approximating optical flow within the MPEG-2 compressed domain," *IEEE-CSVT*, vol. 15, no. 1, pp. 103–107, Jan. 2005.

[4] D. J. Dailey, F. W. Cathey, and S. Pumrin, "An algorithm to estimate mean traffic speed using uncalibrated cameras," *IEEE Trans. Intell. Transp. Syst.*, vol. 1, no. 2, pp. 98–107, Jun. 2000.

[5] M. Isard and A. Blake, “Condensation — conditional density propagation for visual tracking,” *International Journal of Computer Vision*, vol. 29, no. 1, pp. 5–28, 1998.

[6] A. Doucet, N. Gordon, and V. Krishnamurthy, “Particle filters for state estimation of jump Markov linear systems,” *IEEE Trans. Signal Proc.*, vol. 49, no. 3, pp. 613–624, Mar. 2001.

[7] J. Liu and M. West, “Combined parameter and state estimation in simulation-based filtering,” in *Sequential Monte Carlo Methods in Practice*, A. Doucet, N. de Freitas, and N. Gordon, Eds. Berlin: Springer-Verlag, 2001, pp. 197–223.

[8] V. H. Poor, *An introduction to signal detection and estimation*, 2nd ed. Berlin: Springer-Verlag, 1994.

[9] R. Martinez-Cantin, N. de Freitas, and J. A. Castellanos, “Marginal-SLAM: A convergent particle method for simultaneous robot localization and mapping,” in *Proc. IEEE Int. Conf. on Robotics and Automation, ICRA’2007*, Rome, Italy, Apr. 2007.

[10] H.-H. Nagel. Image sequence server. Institut fur Algorithmen und Kognitive Systeme. [Online]. Available: http://i21www.ira.uka.de/image_sequences/index.html