Target Tracking within a Binary Sensor Network

Adrien Ickowicz
IRISA/CNRS, 35042, Rennes, France

J.-Pierre Le Cadre,
IRISA/CNRS,35042, Rennes,France

Abstract

The aim of this paper is to present a new algorithm for target tracking within a binary sensor network. The present work is based on our previous results developed in [1]. A novel tracking method is proposed and its performance through a very classical trajectory model is evaluated. For a given target, this algorithm provides an estimation of its velocity and then of its position. The greatest improvements are made through a position correction and velocity analysis.

1. INTRODUCTION

Sensor networks are systems made of many small and simple sensors deployed over an area in an attempt to sense events of interest within that particular area. In general, the sensors have limited capacities in terms of say range, precision, etc. The ultimate information level for a sensor is a binary one, referring to its output. However, it is important to make a distinction according to the nature of this binary information. Actually, it can be related to a $0-1$ information (non-detection or detection) or to relative $\{-,+\}$ motion information. For example, if the sensors are getting sound levels, instead of using the real sound level (which may cause confusion between loud near objects and quieter close objects), the sensor may simply report whether the Doppler frequency is suddenly changing, which can be easily translated in whether the target is getting closer or moving away. Moreover, low-power sensors with limited computation and communication capabilities can only perform binary detection. We could also cite video sensors, with the intuitive reasoning: the target is getting closer if its size is increasing. The need to use that kind of sensor networks leads to the development of a model for target tracking in binary sensor networks.

We consider a sensor network, made with N sensors (e.g. video), with known positions. Each sensor can only gives us a binary $\{-,+\}$ information [2], i.e. whether the target-sensor distance is decreasing $(-)$ or increasing $(+)$. The main problem with that kind of sensors is that it gives us no direct information on the position or the velocity of the target. In a previous article [1], we concentrated on the estimation of the parameters of a deterministic target trajectory, via batch algorithms. In the present article we release the assumption of (piecewise) constant velocity motion, and we try to follow both position and velocity in real time. In particular, it is shown that it is the trajectory "diversity" which renders this possible.

Obviously, tracking a diffusive Markovian target widely differs from the (batch) estimation of deterministic parameters. However, both problems present strong similarities. Indeed, the geometrical properties remains the same at each instant. The paper is organized as follows. Once the target motion model has been introduced, the most important properties we used to perform the tracking are presented. Then, the method which allows us to perform adapted corrections for tracking the target is presented. It is the main contribution of this paper. The performance of the tracking algorithm is illustrated via several simulation results and we conclude on further works about the tracking in binary sensor networks.

2. TRACKING WITH BINARY SENSORS

A. Target Motion Model

The target is assumed to evolve with a Markov motion, given by:

$$\mathbf{x_k}|\mathbf{x_{k-1}} \quad \sim \quad \mathcal{N}(F_k\mathbf{x_{k-1}}, \mathbf{Q_k}) \tag{1}$$

for $k = 1, 2...$ where $\mathcal{N}(\mu, \sigma^2)$ is a gaussian distribution with mean μ and variance σ^2. The starting position is assumed to be unknown.

B. Sensor Measurement Model and Analysis

At each time period, each sensor gives us a $\{+,-\}$ information, meaning that the target is getting closer or moving away. Given all the sensors reports at the time-period t, we can easily define a space where the target is assumed to be at this time-period. This is the fundamental uncertainty we have at t, and the area of this domain is, of course, directly related to the network parameters (sensor number, network geometry, etc.).

C. Velocity Estimation

We can estimate the direction of the target based on the simple information given by the sensors. Obviously, that estimator will only be precise if the number of sensors is significantly great. To perform that estimation, we can use several methods, such as the Projection Pursuit Regression Method, or the Support Vector Machine Method. The SVM method chosen for our algorithm as a most common method, and is presented in the next paragraphs.

1) Binary Sensor Network Observability Properties: Let us denote s_i a sensor whose position is represented by the vector $\mathbf{t}_i$. Similarly, the vector $\mathbf{x}_t$ represents the position vector of the target at the time-period t. Let us denote $d_i(t)$ the (time-varying) distance from sensor s_i to the target at time t. Then, we have that:

$$d_i(t) \ \searrow \Longleftrightarrow \dot{d}_i(t) < 0 \text{ , or: } \langle \mathbf{x}_t - \mathbf{t}_i, \mathbf{v}_t \rangle < 0 \ , \qquad (2)$$

where $\mathbf{v}_t$ is the instantaneous target velocity. We thus have the following lemma.

Lemma 1: Let s_i (resp. s_j) a sensor whose target distance is decreasing (resp. increasing) at the time-period t, then we have:

$$\langle \mathbf{t}_j, \mathbf{v}_t \rangle \ < \ \langle \mathbf{x}_t, \mathbf{v}_t \rangle \ < \ \langle \mathbf{t}_i, \mathbf{v}_t \rangle \ . \qquad (3)$$

If we restrict to binary motion information, we consider that the output $s_i(t)$ of a sensor (at time t) is $+1$ or -1 according to the distance $d_i(t)$ is decreasing or increasing, so that we have:

$$\begin{cases} s_i(t) = +1 \ \text{ if } \ \dot{d}_i(t) < 0 \ , \\ s_j(t) = -1 \ \text{ if } \ \dot{d}_j(t) > 0 \ . \end{cases} \qquad (4)$$

Let us denote A the subset of sensor whose output is $+1$ and B the subset of sensors whose output is -1, i.e. $A = \{\mathsf{s}_i |\ s_i(t) = +1\}$ and $B = \{\mathsf{s}_j |\ s_j(t) = -1\}$ and $C(A)$ and $C(B)$ their convex hulls, then the following property holds[2]:

Proposition 2: $C(A) \bigcap C(B) = \emptyset$ and $\mathbf{x}_t \notin C(A) \bigcup C(B)$.

Proof: The proof is quite simple and is reproduced here only for the sake of completeness. First assume that $C(A) \bigcap C(B) \neq \emptyset$, this means that there exists an element of $C(B)$, lying in $C(A)$. Let s be this element (and $\mathbf{t}$ its associated position), then we have ($t \in C(B)$):

$$\mathbf{t} = \sum_{j \in B} \beta_j \ \mathbf{t}_j \ , \ \beta_j \geq 0 \text{ and } \ \sum_{j \in B} \beta_j = 1$$

so that we have on the first hand:

$$\langle \mathbf{t}, \mathbf{v}_t \rangle = \sum_{j \in B} \beta_j \ \langle \mathbf{t}_j, \mathbf{v}_t \rangle \ < \ \langle \mathbf{x}_t, \mathbf{v}_t \rangle \ \text{ (see eq. 3)},$$

and, on the other one ($t \in C(A)$):

$$\langle \mathbf{t}, \mathbf{v}_t \rangle = \sum_{i \in A} \alpha_i \ \langle \mathbf{t}_i, \mathbf{v}_t \rangle \ \geq \left(\sum_{i \in A} \alpha_i \right) \ \min_i \ \{ \langle \mathbf{t}_i, \mathbf{v}(t) \rangle \} > \langle \mathbf{x}_t, \mathbf{v}_t \rangle \ . \qquad (5)$$

Thus a contradiction which shows that $C(A) \bigcap C(B) = \emptyset$. For the second part, we have simply to assume that $\mathbf{x}(t) \in C(A)$ ($\mathbf{x}_t = \sum_{i \in A} \alpha_i \ \mathbf{t}_i, \ \alpha_i \geq 0$), which yields:

$$\langle \mathbf{x}_t, \mathbf{v}_t \rangle = \sum_{i \in A} \alpha_i \ \langle \mathbf{t}_i, \mathbf{v}_t \rangle \geq \min_{i \in A} \langle \mathbf{t}_i, \mathbf{v}_t \rangle, \qquad (6)$$

which is clearly a contradiction, idem if $X(t) \in C(B)$. □□□

So, $C(A)$ and $C(B)$ being two disjoint convex subsets, we know that there exists an hyperplane (here a line) separating them. Then, let s_k be a generic sensor, we can write $\mathbf{t}_k = \lambda \ \mathbf{v}_t + \mu \ \mathbf{v}_t^{\perp}$, so that:

$$\langle \mathbf{t}_k, \mathbf{v}_t \rangle = \lambda \left\| \mathbf{v}_t \right\|^2 > 0 \Longleftrightarrow \lambda > 0 \ . \qquad (7)$$

This means that the line spanned by the vector $\mathbf{v}_t^{\perp}$ separates $C(A)$ and $C(B)$. Without considering the translation and considering again the $\left\{ \mathbf{v}_t, \mathbf{v}_t^{\perp} \right\}$ basis , we have :

$$\begin{cases} \mathbf{t}_k \in A \Longleftrightarrow \lambda \left\| \mathbf{v}_t \right\|^2 > \langle \mathbf{x}_t, \mathbf{v}_t \rangle \ , \\ \mathbf{t}_k \in B \Longleftrightarrow \lambda \left\| \mathbf{v}_t \right\|^2 < \langle \mathbf{x}_t, \mathbf{v}_t \rangle \ . \end{cases} \qquad (8)$$

Thus in the basis $(\mathbf{v}_t, \mathbf{v}_t^{\perp})$, the line passing by the point $\left(\dfrac{\langle \mathbf{x}_t, \mathbf{v}_t \rangle}{\left\| \mathbf{v}_t \right\|^2}, 0 \right)$ and whose direction is given by $\mathbf{v}_t^{\perp}$ is separating $C(A)$ and $C(B)$.

2) The Support Vector Machine (SVM) approach [3]: As seen previously, the problem we have to face is to optimally separate the two classes of sensors (i.e. the $+$ and $-$). So, we can use the general framework of SVM, widely used in the classification context. The set of labeled patterns $\{ (y_1, \mathbf{x}_1), \cdots, (y_l, \mathbf{x}_l) \}$ ($y_i \in \{-1, 1\}$ and $\mathbf{x}_i$ sensor positions) is said to be linearly separable if there exists a vector $\mathbf{w}$ and a scalar b such that the following inequalities hold true:

$$\begin{cases} \langle \mathbf{w}, \mathbf{x}_i \rangle + b \geq 1 \quad \text{if} : y_i = 1 \ , \\ \langle \mathbf{w}, \mathbf{x}_i \rangle + b \leq -1 \quad \text{if} : y_i = -1 \ . \end{cases} \qquad (9)$$

Let $\mathcal{H}(\mathbf{w}, b) \triangleq \{ \mathbf{x} | \langle \mathbf{w}, \mathbf{x} \rangle + b = 0 \}$ ($\mathbf{w}$: normal vector) be this optimal separation plane and define the margin (marg) as the distance of the closest point $\mathbf{x}_i$ to $\mathcal{H}$, then it is easily seen that $\mathsf{marg} = \frac{1}{\|\mathbf{w}\|}$. Thus, maximizing the margin lead to consider the following problem:

$$\left| \begin{array}{l} \min\limits_{\mathbf{w}, b} \ \tau(\mathbf{w}) \stackrel{\delta}{=} \left\| \mathbf{w} \right\|^2 \ , \\ \text{s.t. } : y_i \left(\langle \mathbf{w}, \mathbf{x}_i \rangle + b \right) \geq 1 \ \ \forall \, i = 1, \cdots, l \ \ y_i = \pm 1 \ . \end{array} \right. \qquad (10)$$

Denoting Λ the vector of Lagrange multipliers, dualization of eq. 10 leads to consider again a quadratic problem, but with more explicit constraints [3], i.e. :

$$\left| \begin{array}{l} \max\limits_{\Lambda} \ W(\Lambda) = -\dfrac{1}{2} \ \Lambda^T \ D \ \Lambda + \Lambda^T \ \mathbf{1} \ , \\ \text{s.t. } : \Lambda \geq 0 \ , \ \Lambda^T Y = 0 \ , \end{array} \right. \qquad (11)$$

where $\mathbf{1}$ is a vector made of 1 and $Y^T = (y_1, \cdots, y_l)$ is the l-dimensional vector of labels, and D is the Gram matrix:

$$D_{i,j} = \langle y_i \mathbf{x}_i, y_j \mathbf{x}_j \rangle \ . \qquad (12)$$

The dualized problem can be efficiently solved by classical quadratic programming methods. The less-perfect case consider the case when data cannot be separated without

errors and lead to replace the constraints of eq. 10 by the following ones:

$$y_i \left(\langle \mathbf{w}, \mathbf{x}_i \rangle + b \right) \geq 1 - \xi_i \ , \ \xi_i \geq 0 \ , \ i = 1, \cdots, l \ . \quad (13)$$

Consider now a multiperiod extension of the previous analysis. Let us restrict first to a two-period analysis, we shall consider two separating hyperplanes (say $\mathcal{H}_1, \mathcal{H}_2$) defined by:

$$\begin{cases} \langle \mathbf{w}, x_l^1 \rangle + b_1 \gtrless \pm c_1 \ \text{ according to: } \ y_l^1 = \pm 1 \ , \\ \langle \mathbf{w}, x_l^2 \rangle + b_2 \gtrless \pm c_2 \ \text{ according to: } y_l^2 = \pm 1 \ . \end{cases} \quad (14)$$

It is also assumed that these two separating planes are associated with time periods T and $T + \Delta T$, ΔT known. It is easily seen that the margin for the separating plane $\mathcal{H}_1$ is $\frac{c_1}{\|\mathbf{w}\|}$, while for the plane $\mathcal{H}_2$ it is $\frac{c_2}{\|\mathbf{w}\|}$. Thus, the problem we have to solve reads:

$$\left| \begin{array}{l} \min\limits_{\mathbf{w}, c_1, c_2, b_1, b_2} \quad \left[\max\limits_{1,2} \left(\dfrac{\|\mathbf{w}\|^2}{c_1^2}, \dfrac{\|\mathbf{w}\|^2}{c_2^2} \right) \right] \ , \\ \text{s.t.: } \ y_l^1 \ \left(\langle \mathbf{w}, x_l^1 \rangle + b_1 \right) \geq c_1 \ , \ y_l^2 \ \left(\langle \mathbf{w}, x_l^2 \rangle + b_2 \right) \geq c_2 \ \forall l. \end{array} \right. \quad (15)$$

At a first glance, this problem appears as very complicated. But, without restricting generality, we can assume that $c_1 < c_2$. This means that $\max\limits_{1,2} \left(\dfrac{\|\mathbf{w}\|^2}{c_1^2}, \dfrac{\|\mathbf{w}\|^2}{c_2^2} \right) = \dfrac{\|\mathbf{w}\|^2}{c_1^2}$. Making the changes $\frac{1}{c_1} \mathbf{w} \rightarrow \mathbf{w}^{'}$ and $\frac{b_1}{c_1} \rightarrow b_1^{'}$ then leads to consider the classical problem:

$$\left| \begin{array}{l} \min\limits_{\mathbf{w}^{'}, b_1^{'}, b_2^{'}} \ \left\| \mathbf{w}^{'} \right\|^2 \\ \text{s.t. : } \ y_l^1 \ \left(\langle \mathbf{w}^{'}, x_l^1 \rangle + b_1^{'} \right) \geq 1 \ , \ y_l^2 \ \left(\langle \mathbf{w}^{'}, x_l^2 \rangle + b_2^{'} \right) \geq 1 \ \forall l. \end{array} \right. \quad (16)$$

Let $\mathbf{w}^*$ be the (unique) solution of eq. 16, then a straightforward calculation yields the distance $d(\mathcal{H}_1^*, \mathcal{H}_2^*)$ between the two separating planes, i.e.:

$$d(\mathcal{H}_1^*, \mathcal{H}_2^*) = \frac{|b_1^* - b_2^*|}{\|\mathbf{w}^*\|} \ .$$

Finally, we deduce that the estimated velocity vector $\hat{\mathbf{v}}$ is given by:

$$\hat{\mathbf{v}} = \alpha \ \mathbf{w}^* \ \text{ and: } \ \hat{v} = \frac{1}{\Delta T} \ d(\mathcal{H}_1^*, \mathcal{H}_2^*) \ . \quad (17)$$

The previous analysis can be easily extended to an arbitrary number of periods, as long as the target trajectory remains rectilinear. Another definite advantage is that it can be easily extended to multitarget tracking.

3) The effect of target acceleration: To illustrate the effect of velocity change for estimating the target position, let us consider a very simple example. Assume that the target motion is uniformly accelerated, i.e. :

$$\mathbf{x}_t = \mathbf{x}_0 + t \ \dot{\mathbf{x}}_0 + \frac{t^2}{2} \ \ddot{\mathbf{x}}_0 \ . \quad (18)$$

We have now to deal with the following question: Is the target trajectory fully observable? To that aim, we first recall the following result. Considering a dense binary network, two target trajectories are said indistinguishable iff they provide the same (binary) information which is equivalent to the following conditions:

$$\left\{ \ \dot{\mathbf{x}}_t = \dot{\mathbf{y}}_t \ , \langle \mathbf{y}_t - \mathbf{x}_t, \dot{\mathbf{y}}_t \rangle = 0 \ \ \forall t \ . \right. \quad (19)$$

Expliciting the second condition of eq 19, with the target motion model 18, we obtain that the following condition holds ($\forall t$):

$$\begin{array}{l} \langle \mathbf{y}_0 - \mathbf{x}_0, \dot{\mathbf{y}}_0 \rangle + t \langle \dot{\mathbf{y}}_0 - \dot{\mathbf{x}}_0, \dot{\mathbf{y}}_0 \rangle + \frac{1}{2} t^2 \langle \ddot{\mathbf{y}}_0 - \ddot{\mathbf{x}}_0, \dot{\mathbf{y}}_0 \rangle \ , \\ + t \langle \mathbf{y}_0 - \mathbf{x}_0, \ddot{\mathbf{y}}_0 \rangle + t^2 \langle \dot{\mathbf{y}}_0 - \dot{\mathbf{x}}_0, \ddot{\mathbf{y}}_0 \rangle + \frac{1}{2} t^3 \langle \ddot{\mathbf{y}}_0 - \ddot{\mathbf{x}}_0, \ddot{\mathbf{y}}_0 \rangle = 0 \ . \end{array} \quad (20)$$

Thus, $\langle \mathbf{y}_t - \mathbf{x}_t, \dot{\mathbf{y}}_t \rangle$ is a zero polynomial, which means that all its coefficients are zero. For the t^3 coefficients we obtain the condition $\langle \ddot{\mathbf{y}}_0 - \ddot{\mathbf{x}}_0, \ddot{\mathbf{y}}_0 \rangle = 0$. Similarly with the $\langle \mathbf{y}_t - \mathbf{x}_t, \dot{\mathbf{x}}_t \rangle = 0$ condition, we obtain $\langle \ddot{\mathbf{y}}_0 - \ddot{\mathbf{x}}_0, \ddot{\mathbf{x}}_0 \rangle = 0$. Subtracting these two equalities yield $\| \ddot{\mathbf{y}}_0 - \ddot{\mathbf{x}}_0 \| = 0$, or $\ddot{\mathbf{x}}_0 = \ddot{\mathbf{y}}_0$.

Quite similarly, we obtain the equality $\dot{\mathbf{x}}_0 = \dot{\mathbf{y}}_0$ and the last equality:

$$\langle \mathbf{y}_0 - \mathbf{x}_0, \dot{\mathbf{y}}_0 + t \ddot{\mathbf{y}}_0 \rangle = 0 \ \ \forall t \ . \quad (21)$$

Assuming that the couple $\{ \dot{\mathbf{y}}_0, \ddot{\mathbf{y}}_0 \}$ spans the sensor space then we deduce that $\mathbf{x}_0 = \mathbf{y}_0$. So, it has been shown that it was the target acceleration which render the problem fully observable. This reasoning can be extended to a wide variety of target modeling.

3. TARGET TRACKING

The main issue with the SVM estimation is that it only provides us the general direction of the target within a deterministic framework. Moreover, it is highly desirable to develop a reliable algorithm for target tracking (velocity and position). To solve this problem, we build a two-step algorithm. In the first step, we perform a correction through the estimated unitary velocity vector at each time-period t, called λ_t. Then, in a second time, we perform a correction through the orthogonal-estimated (unitary) velocity vector, also at each time-period, called θ_t. These two corrections give us a better estimation of both the velocity and the position of the target. We refer to fig. 1 for the presentation of the rationale of the two correction factors.

A. The λ factor

To build that correction factor, we started with a very simple assumption. At each period t, the sensors provide binary motion information. Thanks to the first part of this article, we know that the target is in the (special) set lying between the two same-sign-sensors set. Then, starting from the previous estimated position of the target, we move the estimated target through the estimated velocity vector direction until it stands in that special set. We now define this operator in a mathematical way:
Let $\hat{\mathbf{v}}_t$ be the estimated normalized velocity vector at time t.

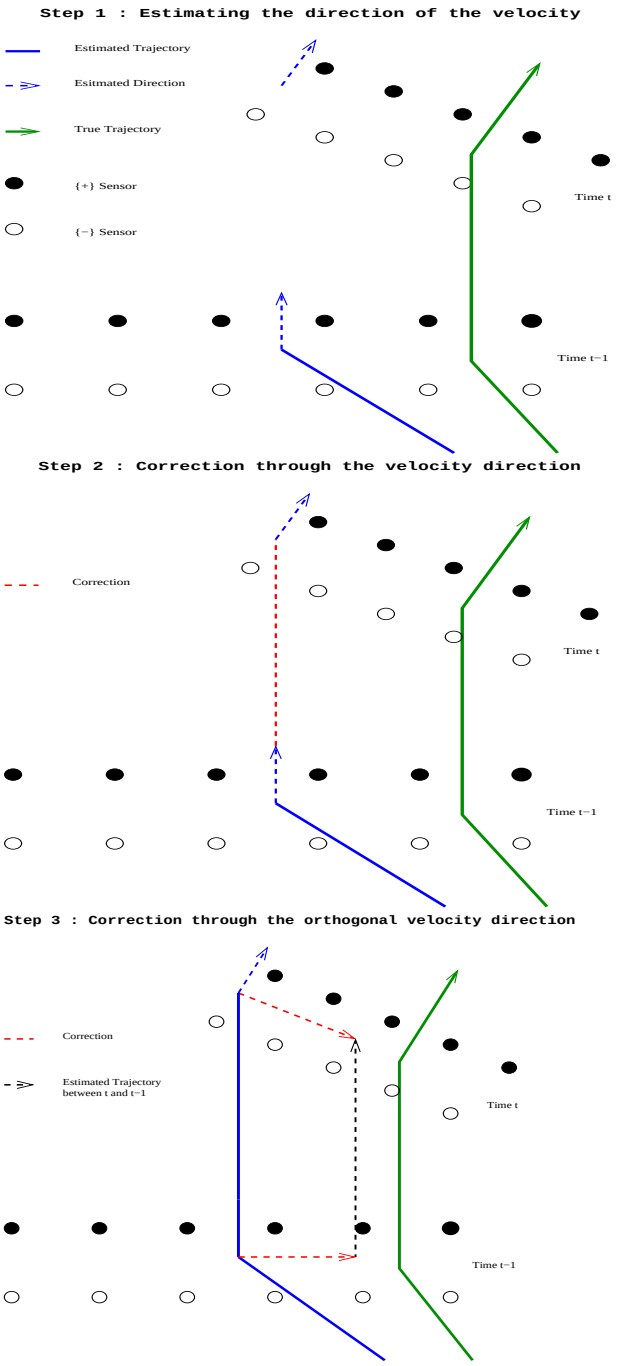

Fig. 1: *Correction scenario.*

Moreover, let $\{\mathbf{t}_i^{(-)}\}_i$ (respectively $\{\mathbf{t}_i^{(+)}\}_i$) be the coordinates of the sensors (s_i) giving a $\{-\}$ (respectively a $\{+\}$) at time t.
We sort $vs_i^{(-)} = \langle \hat{\mathbf{v}}_t, \mathbf{t}_i^{(-)} \rangle$ (respectively $vs_i^{(+)} = \langle \hat{\mathbf{v}}_t, \mathbf{t}_i^{(+)} \rangle$). Then, following a very simple geometrical reasoning, we note that $\langle \hat{\mathbf{v}_t}; \hat{X}_t \rangle$ should be between $vs_{max}^{(-)}$ and $vs_{min}^{(+)}$. To ensure that property, we define the following correction factor:

$$\lambda_t = \frac{vs_{moy}^{(+,-)} - \langle \hat{\mathbf{v}}_t, \hat{\mathbf{x}}_{t-1} \rangle}{\langle \hat{\mathbf{v}}_t, \hat{\mathbf{v}}_{t-1} \rangle} \ , \quad \text{with the following definition of } vs_{moy}^{(+,-)} : \quad vs_{moy}^{(+,-)} = \frac{vs_{max}^{(-)} + vs_{min}^{(+)}}{2} \ , \tag{22}$$

To calculate this factor, we consider the projection equality:

$$\langle \hat{\mathbf{v}}_t, (\hat{\mathbf{x}}_{t-1} + \lambda_t \ \hat{\mathbf{v}}_{t-1}) \rangle = vs_{moy}^{(+,-)} \tag{23}$$

which means that the projection of the corrected value is equal to the mean value of the projection. Geometrically, this means that the position of the target is estimated to be in the center of the special set defined by the sensors. The value of the correction factor λ_t (see eq. 22) is then straightforwardly deduced from eq. 23. Similarly, the target position is updated via:

$$\hat{\mathbf{x}}_t^{corr} = \hat{\mathbf{x}}_{t-1} + \lambda_t \ \hat{\mathbf{v}}_{t-1} \ . \tag{24}$$

Here the correction factor λ_t has been calculated via the average value of the projection. This is an arbitrary choice and we can consider the lower or the upper bound of the projection with no significant difference on the results of the algorithm.
Obviously, if the estimation of the position is not very good, the estimated velocity value (clearly based on λ_t) will be quite different from the real value of the velocity. The next correction factor is based on the assumption that the target velocity changes are upper and lower bounded.

B. The θ correction factor

We assume that the velocity of the target has bounded acceleration. Then, if the velocity estimated at a certain time t is too different from the velocity estimated at time $t-1$, this means that the estimated position of the target is far from the right one. Then, in that precise case, we consider an orthogonal correction, through $\hat{\mathbf{v}_\mathbf{t}}^{\perp}$.
For that deterministic algorithm we decided to perform a very simple modeling of the velocity. Indeed, we take as a right value for the velocity the simple mean of the k previous values of the estimated velocity ($m_{t,k}$). We calculate in addition the variance ($\sigma_{t,k}$), and the factor θ_t can be non-zero iff the estimated value of the velocity at time t is not in the interval given by $[m_{t,k} - \sigma_{t,k}; m_{t,k} + \sigma_{t,k}]$. We then look for θ_t such that:

$$\langle \hat{\mathbf{x}}_t^{corr} + \theta_t \ \hat{\mathbf{v}}_t^{\perp} - (\hat{\mathbf{x}}_{t-1} + \theta_t \ \hat{\mathbf{v}}_{t-1}^{\perp}); \hat{\mathbf{v}}_{t-1} \rangle = m_{t,k} \ . \tag{25}$$

The previous equation needs some explanation. Given that $\hat{\mathbf{x}}_t$ is the estimated target position at time t, we would like to correct the value to be closer to the right position. The only way we can deal with it, is to correct the estimated value of the velocity. $\hat{\mathbf{x}}_t^{corr} - \hat{\mathbf{x}}_{t-1}$ is the previous calculated correction. If the difference between that estimation and the value $m_{t,k}$ is too important, we try to reduce that difference with a translation of the positions at time periods t and $t-1$. As we want the positions to stay in the special set defined by the sensors, the direction

of that translation is given by $\hat{\mathbf{v}}_t^{\perp}$ for the position at time t, and $\hat{\mathbf{v}}_{t-1}^{\perp}$ for the position at time $t-1$.
Performing straightforward calculation, leads to consider the following correction factor:

$$\theta_t = \frac{m_{t,k} - \lambda_t}{\langle \hat{\mathbf{v}}_t^{\perp}; \hat{\mathbf{v}}_{t-1} \rangle} \ . \tag{26}$$

Obviously, as we could expect when presenting the method, if the target motion is rectilinear and uniform , no correction factor can be calculated. Then, the final estimated position is given by:

$$\hat{\mathbf{x}}_t^{fin} = \hat{\mathbf{x}}_t^{corr} + \theta_t \ \hat{\mathbf{v}}_t^{\perp} \ . \tag{27}$$

C. The final correction step

Noticeably the most important step of the algorithm, i.e. the θ correction factor, is based on the estimation of the velocity change. Indeed, the best the estimation of the velocity is, the best we can estimate the position. Then, our aim is to perform a better analysis of the target motion. Considering that from time to time, the estimation of the position increases in quality, a promising way should be to perform a feedback of the newest corrector to the oldest position estimation. We denote $\hat{\mathbf{z}}_t$ the updated estimated position of the target at time t. Then, according to the previous paragraph, the estimated position is updated via:

$$\forall j < t: \ \ \hat{\mathbf{z}}_j = \hat{\mathbf{x}}_j^{fin} + \sum_{i=j+1}^{t} \theta_i \hat{\mathbf{v}}_i^{\perp} \ . \tag{28}$$

With this new estimator we will be able to perform a better analysis of the target motion (position and velocity).

D. The tracking algorithm

With the definition of the correction factors, the theoretical part of the algorithm is finished. Then, it is presented as follows, at time period t:

1) Get the binary information of each sensor, and then the target position set.
2) Estimate the velocity direction at time t via a SVM method
3) Perform the λ calculation, and add that correction to the estimated velocity at time $t-1$. The time-t-position is then updated.
4) Check if the estimated velocity at time $t-1$ is too different from the modeled value, and in this case, calculate θ_t.
5) Update the position at time t, and in this case, the velocity at time $t-1$ with the correction θ.

Steps 2 and 3 can be inverted with no damage in the process. This is the main part of the algorithm. However, there is no mention in that enumeration of the initialization. There are two main state vectors that have to be initialized. The position and the velocity. The position is assumed to be unknown, but thanks to the sensors, we can have a space where the target is assumed to be at first. We use here a uniform law for the initialization, given that we have no further information about where the target can start.
The initialization of the velocity is not far from that solution. Indeed, with the binary information, we can provide a convenient estimate of the velocity direction. Even if we don't have a precise idea of the speed value, we can then start the algorithm.

4. Simulation Results

We will present in that section the results of the tracking algorithm. We consider here that the target starts from the $[100, 100]$ position and that its initial velocity vector is the $[1, 1]$ vector. The number of sensors is equal to 70, in a quite wide space (300mx300m). The variance of the target motion is not very important, and the tracking duration is $T = 30$ seconds.
One simulation is presented in figure 2. In red is represented the real target trajectory, quite diffusive, and in green the estimated successive positions. The initialization is not very bad because the number of sensors is quite important, which means that the uniform set is not too large. After the first step, the estimation seems to hang the real trajectory, and follows the target well (less than 10 meter error). However, when the target turns right, we loose some precision, mainly because the correction factors seemed to be "lost". The reason for that behavior is that the SVM method provides us a bad estimation of the velocity vector. Then, the algorithm provides a correction in a bad direction, which moves away from the real trajectory. During a few seconds, the estimation works quite bad, before hanging again the target direction, and then performing a quite good estimation of the velocity. Unfortunately, there is no evidence in that example that increasing indefinitely the tracking duration results in an estimated position closer and closer to the real target position. This is precisely the aim of the two next figures,

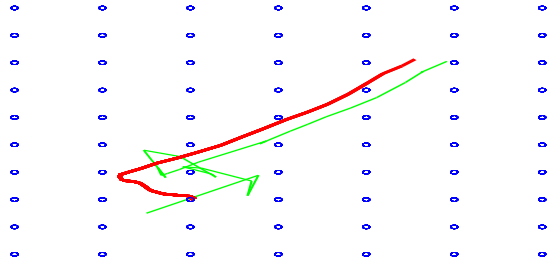

Fig. 2: *Trajectory Estimation of a target. In Red, the real trajectory, in green the estimated one.*

3 and 4. The first one shows the mean square error of the estimated position of the target through the trajectory. The total time is $T = 30$ seconds, and we can see an amazing

and remarkable decrease of that MSE in the first seconds. It seems however that there is a limit to that decrease. Indeed, the MSE will not converge to a zero value, even if we could perform a long-time tracking. Clearly, the limitation is due to the binary information at first, and certainly to the number of sensors in a second time. Some further work could certainly exhibits a strong link between the number of sensors and the MSE of the position.

In the same way, the velocity estimation has some

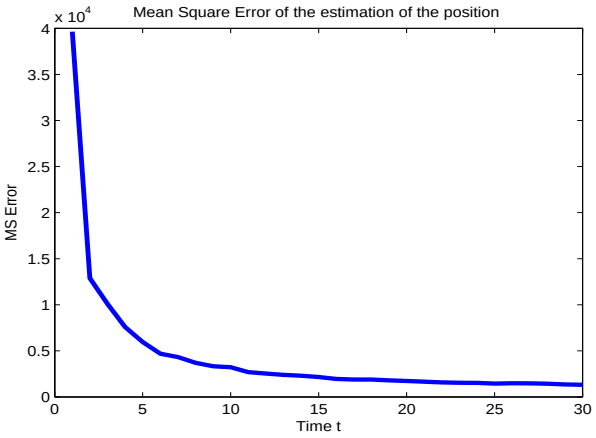

Fig. 3: *MSE of the estimator of the target position.*

acceptable MSE through out the tracking process. Despite the clearly strong peak at the beginning, the curve then stands to an acceptable but non zero value. The effect is more obvious than in the position case, surely because of the velocity modeling we make in the algorithm, which forces the velocity estimation to very bad evolution. A clue could be to perform a most sophisticated modeling of the velocity, but given the binary information, this won't be easy. This is another work in progress for the evolution of our algorithm.

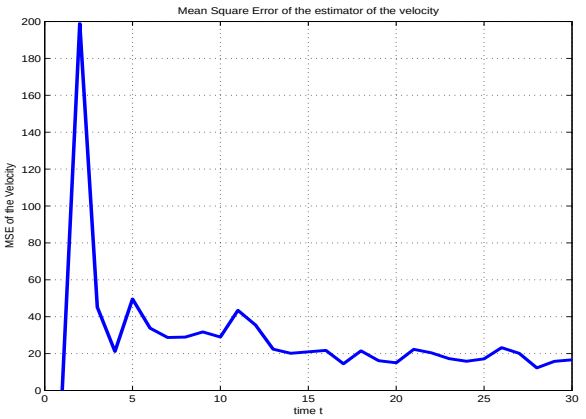

Fig. 4: *MSE of the estimator of the target velocity.*

5. Conclusion

A new method for tracking both position and velocity of a moving target via binary data has been developed. Though the instantaneous data are poorly informative, our algorithm takes benefit of the network extent and density via specific spatio-temporal analysis. This is remarkable since the assumptions we made about target motion are not restrictive. Noticeably also, our algorithm is quite fast and reliable. Furthermore, it is clear that performance can be greatly improved if we can consider that the acquisition frequency is (far) greater than the maneuver frequency. In particular, we can mix the present method with the one we developed in [1].

However, some important questions remain. The first one concerns the velocity modeling. We focused on this paper on the adaptability of the different correction factors, but we didn't pay much attention to that modeling, which can definitely improve the estimation quality. Moreover, our tracking algorithm is basically deterministic even if the target motion modeling is basically probabilistic. Thus, it should be worth to calculate the first correction factor (λ) via a likelihood, such that $\mathbf{x}^{corr}$ does not always stays in the mean of the special set. Moreover, that likelihood should be related to all the sources of sensor uncertainty. In addition, the present algorithm gives a slow response to sudden target maneuver. A remedy should be to incorporate a stochastic modeling of such event in our algorithm.

The second correction factor (θ) may also be improved via a stochastic approach. Instead of considering a correction only related to the estimated velocity estimated, we could immerse this correction within a stochastic framework involving both $\hat{\mathbf{v}}_t$ and θ. These observations are part of our next work on that very constrained but also quite exciting tracking framework. The last important point is multiple target tracking. Even if our work in this area is quite preliminary, it is our strong belief that our spatio-temporal separation based algorithm should be the natural way to overcome the association problems.

References

[1] A. Ickowicz, J.-P. Le Cadre, A new method for target trajectory estimation within a binary sensor network. *Proc. of the 10th European Conference on Computer Vision: Multi-camera and Multi-modal Sensor Fusion Algorithms and Applications Workshop*, Oct 2008.

[2] J. Aslam, Z. Butler, F. Constantin, V. Crespi, G. Cybenko, D. Rus, Tracking a moving object with a binary sensor network. *Proc. of the 1st international Conference on Embedded Networked Sensor Systems*, Nov 2005, pp. 150–161.

[3] C. Cortes, V. Vapnik, Support-Vector Networks. Machine Learning, **20**, 1995, pp. 273–297.

[4] J.H. Friedman and J. H. Tuckey, A Projection Pursuit Algorithm for Exploratory Data Analysis. IEEE Trans. Comput. **23**, 1974, pp. 881–889.

[5] J. H. Friedman and W. Stuetzle, Projection Pursuit Regression. J. Amer. Stat. Soc., **76**, 1981, pp. 817–823.

[6] L. Lazos, R. Poovendran and J.A. Ritcey , Probabilistic Detection of Mobile Targets in Heterogeneous Sensor Networks. *Proc. of the 6-th IPSN*, Apr. 2007.

[7] X. Wang and B. Moran , Multitarget Tracking Using Virtual Measurements of Binary Sensor Networks. *Proc. of the 9-th Int. Conf. on Information Fusion*, Jul. 2006.