Bi-target Tracking within a Binary Sensor Network

Adrien Ickowicz
IRISA / CNRS
Rennes, FRANCE.
ickowicz@irisa.fr

J.-P. Le Cadre
CNRS
Rennes, FRANCE
lecadre@irisa.fr

Abstract – *The aim of this paper is to present an algorithm for the tracking of two targets moving through a binary sensor network. We previously developed a new deterministic algorithm for the purpose with only one target, and we now provide a improved version of that algorithm, considering an association problem. In addition, we present some interesting results on the mean square error of the position estimation.*

1 Introduction

Sensor networks are systems that can be 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.

There are several limitations in the use of such binary information, but we demonstrated that a rather good estimation of both position and velocity of a target could be performed by the use of a well-selected spatio-temporal information. Even if the results where only obtained with a single target, we will demonstrate in that paper that by adding two prior steps to our algorithm, two targets can be tracked by the binary sensor network. In the first part of the article, we will recall all the issues that can be encountered with only a single target, and what solution we previously proposed. We will then present the new problem we try to face, and the algorithm developed for that purpose. We will continue with the presentation of simulation results before we conclude on the efficiency of our solution.

2 Tracking with Binary Sensors

2.1 Target Motion Model

In the first paper [1], the target was 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 was assumed to be unknown. Considering two targets for the main purpose of that article, we decided that both of them will have a trajectory that can be modeled as defined in eq. 1

2.2 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 a time period t, and the area of this domain is, of course, directly related to the network parameters (sensor number, network geometry, etc.).

2.3 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 is the most common

statistical-used method for classification, and is presented in the next paragraphs.

2.3.1 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:

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

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:

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 ($t \in C(A)$):

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

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 :

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

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 $C(B)$.

2.3.2 The Support Vector Machine (SVM) approach

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:

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

Let $\mathcal{H}(\mathbf{w}, b) \stackrel{\Delta}{=} \{\mathbf{x} | \langle \mathbf{w}, \mathbf{x} \rangle + b = 0\}$ ($\mathbf{w}$: 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, 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:

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

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:

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

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 $\max\limits_{1,2} \left(\frac{\|\mathbf{w}\|^2}{c_1^2}, \frac{\|\mathbf{w}\|^2}{c_2^2} \right) = \frac{\|\mathbf{w}\|^2}{c_1^2}$. 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:

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

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.

2.3.3 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. :

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

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:

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

Explicating 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, $\langle \mathbf{y}_t - \mathbf{x}_t, \dot{\mathbf{x}}_t \rangle$ 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 $\langle \mathbf{y}_t - \mathbf{x}_t, \dot{\mathbf{x}}_t \rangle = 0$, 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.

2.4 Estimation of both velocity and position

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 built a two-step algorithm. In the first step, we performed a correction through the estimated unitary velocity vector at each time-period t, called λ_t. Then, in a second time, we performed a correction through the orthogonal-estimated (unitary) velocity vector, also at each time-period, called θ_t. These two corrections gave 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.

2.4.1 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

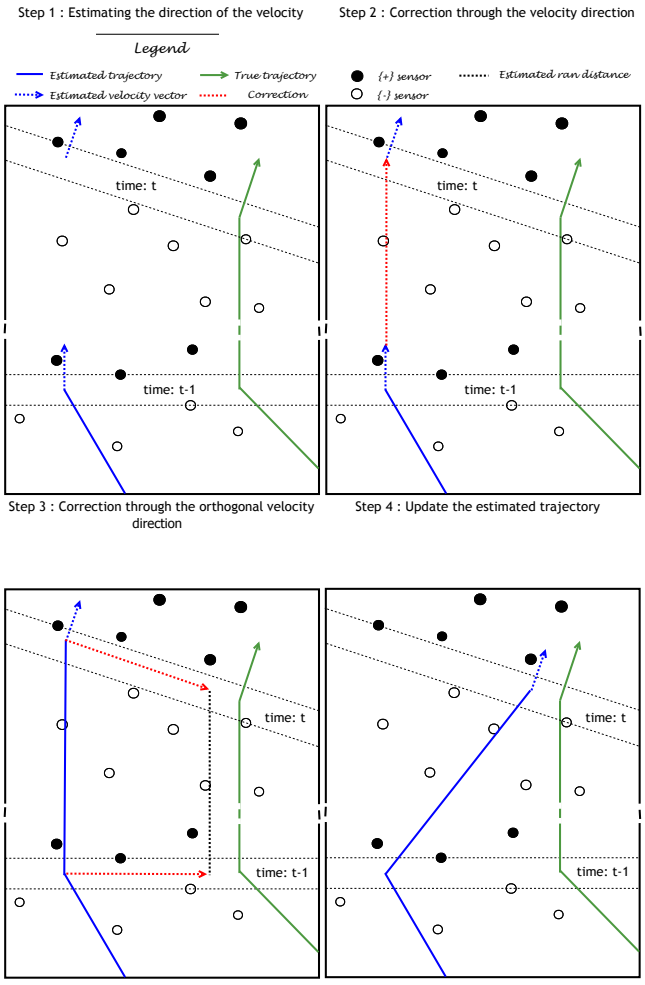

Figure 1: *Correction scenario.*

target through the estimated velocity vector direction until it stands in that special set. We now define this operator in a mathematical way:

Let $\hat{\mathbf{v}}_t$ the estimated normalized velocity vector at time t.

Moreover, let $\{\mathbf{t}_i^{(-)}\}_i$ (respectively $\{\mathbf{t}_i^{(+)}\}_i$) 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 $\langle \hat{\mathbf{v}}_\mathbf{t}; \hat{X}_t \rangle$ 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.

2.4.2 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:

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

2.4.3 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:

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

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

3 Track two targets

We want to use our first work as a real support to the multi-target tracking. We then are back to association problems that are solved with different algorithms. However, it is not exactly a classical association problem given that we consider that each sensor can give us the number of targets getting closer, but not which targets are getting closer. Then, the first step is to classify the scenario. The last difference with the one-target case is that the sensors are supposed reliable, and no false information is processed.

3.1 Classification of the situation

Considering the set of sensors as a closed convex space, there are only two possibilities for velocity-orthogonal straight lines of the targets. Whether the have a common point, whether they don't. These two particular situations needs two different approaches.

3.1.1 Crossing lines

If the two lines have a common point, this means we will have theoretically four different kinds of information, but practically three. We have for each sensor $\{+,+\}$, $\{-,-\}$ or $\{+,-\} = \{-,+\}$. Given that a $+$ gives to each sensor counter a $+1$ and a $-$ gives 0, we have three different sets: A_0, A_1 and A_2. Both targets are getting closer to A_2, and are moving away from A_0. The aim of that part is then to determine the all 4 different sets given only 3 of them.

To find the 4 different sets, we purpose to start by separating A_0 and A_2. But instead of using the classical SVM classification, the equation we want to solve is:

$$min_i[y_i(\langle \mathbf{w}, \mathbf{x}_i \rangle + b)] \quad = \quad 0 \tag{29}$$

where $\mathbf{w}$ and b are unknown. As a matter of fact, solving this equation will lead to a line separating A_0 and A_2, and will also separate A_1 into two different sets, which appear to be on one hand $\{+,-\}$ (A_1^+) and on the other hand $\{-,+\}$ (A_1^-).

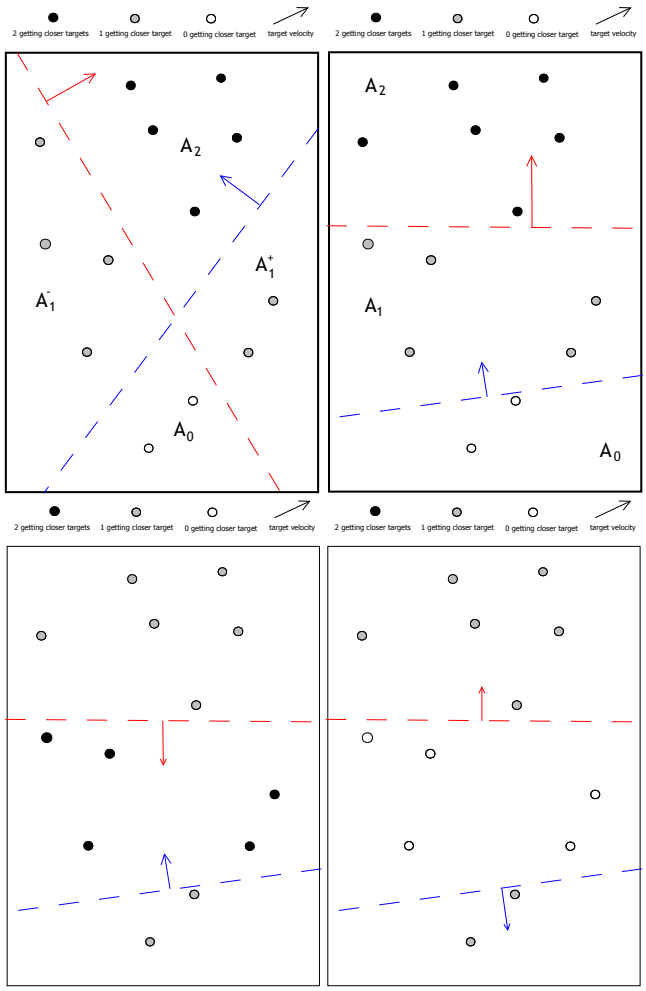

Figure 2: *Situation 1 (on the top left hand), Situation 2 (on the top right hand) ans Situation 3 (On the bottom hand).*

Proposition 3 *The straight line defined by eq. 29 converges as the number of sensors grows to the right line, and then to the right separation of A_1^+ and A_1^-.*

Proof: Let be **L** the straight line defined by eq. 29. As the number of sensors grows, the distance between the right line and A_0 and A_2 decreases to 0. See [2] for a more complete proof.
□□□

We then have the four different sets, that we can now easily separate ($A_2 \bigcup A_1^+$ vs $A_0 \bigcup A_1^-$ and $A_2 \bigcup A_1^-$ vs $A_0 \bigcup A_1^+$) with the SVM method.

3.1.2 No-crossing lines

There exist two different situations for that purpose. Whether the two targets are moving through the same direction, whether they are facing. In the first case, you just have to separate (still with the SVM) A_0 from A_1 and then A_1 from A_2. The second case is a little bit more complicated.
We will consider the cases presented in fig. 2 for simplicity.

Our main problem is here to discriminate A_1^l and A_1^r. We will perform this by building a straight line separating the two. A_0 is a convex hull, and the straight line crossing all that set will separate A_1^l and A_1^r. As the sensor network is known, we also have the knowledge of the border-sensors positions. Then, we take each of these, and we separate the zeros and the ones. Given the convex property of each set, we have the insurance that the sequence of sensors will give us two chains of zeros, separated by two series of ones. Then, taking one sensor in each zero-zone, and building the straight line joining the two sensors, we have the wanted separation.

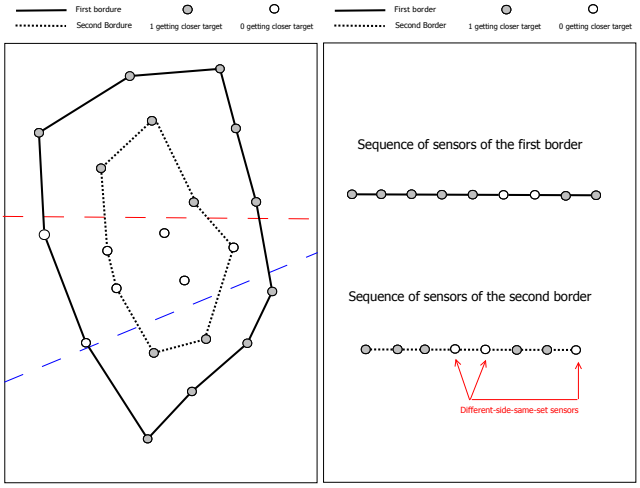

Figure 3: *Situation 3: Trouble with the first border.*

However, there can be a case that cannot be solved. Indeed, we can assume the case in fig. 3. We can see that the first border (i.e. the external) has only one sequence of zeros. We then have to perform the second border, then the next one, until we get two sequences of zeros separated by sequences of ones. Obviously, the bigger the number of border we need to calculate is, the biggest care we have to take by building the line. Indeed, the line drawn by joining two inside sensors can leave the convex hull. Then the separation appears to be less reliable.

3.1.3 Decision Rule

As described, there are not two different situations, but three, with three different solutions. Then, to perform our tracking, we need to decide in which case we are.
The first decision rule to be taken, is the difference between the greater encounter and the lower one. If the difference is 2, we then are in the third described situation. If it's 3, we are in situation 1 or 2. Unfortunately, we haven't been able yet to find an efficient decision rule to classify the two situation. The solution we then chose is based on the fact that the sensors are perfect. We separate A_0 from A_1 and A_1 from A_2, and if no misclassification appears, we are in case 2. If there are errors, we will consider situation 1.

Algorithm 1 Binary Sensor Network Bi-Target Tracking Algorithm

Require: $\forall t, S_t$
$\hat{X}_0 \sim \mathcal{U}(S_0)$
$\hat{V} \leftarrow \mathcal{N}(m, \sigma^2)$
for $t = 1$ to T **do**
  $S_t \leftarrow$ Decision Rule
  $(\mathbf{w}_{i,t}, \hat{\mathbf{v}}_t, \hat{\mathbf{x}}_t) \leftarrow SVM(S_t)$
  $\lambda_{(ij,t)} \leftarrow \frac{vs(\hat{V}) - \langle \hat{\mathbf{v}}_t, \hat{\mathbf{x}}_{t-1} \rangle}{\langle \hat{\mathbf{v}}_t, \hat{\mathbf{v}}_{t-1} \rangle}$
  $\theta_{(ij,t)} \leftarrow \frac{|\hat{V}| - \lambda_t}{\langle \hat{\mathbf{v}}_t^{\perp}; \hat{\mathbf{v}}_{t-1} \rangle}$
  $\tilde{p}_{ij} \leftarrow \frac{1}{\sigma\eta 2\pi} e^{-\frac{(\lambda_{ij,t})^2}{\eta^2} - \frac{(\mathbf{w}_{(i,t)} - \mathbf{w}_{(j,t-1)})^2}{\sigma^2}}$
  $(\lambda_t^*, \theta_t^*) \leftarrow \tilde{p_{ij}}$
  $\hat{X}_t \leftarrow \lambda_t^* \hat{v}_t + \theta_t^* \hat{v}_t^{\perp}$
  $\hat{V} \leftarrow \hat{X}_t - \hat{X}_{t-1}$
end for
return $\hat{X}, \hat{V}$

3.2 Association Problem

At a certain time t, we have two trajectories estimated until time $t-1$, and an information for the position of both targets at time t. We then need to know what target can be associated with which trajectory.
We first build an association matrix, such that:

$$\forall (i,j) \in [1,2], \quad p_{ij} = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(\mathbf{w}_{(i,t)} - \mathbf{w}_{(j,t-1)})^2}{\sigma^2}} \tag{30}$$

where $\mathbf{w}_{(i,t)}$ is the SVM-calculated parameter of the separating line, for the target i, at time t.
We then associate the trajectories and the one-time estimation with the greatest probability. However, there can be a mistake if the two targets are moving with the same global behavior. We need then one more verification. We calculate for all the possible associations the λ_t correction. And as we want the correction to be as small as possible, this leads to that expression:

$$\forall (i,j) \in [1,2], \quad \tilde{p}_{ij} = \frac{p_{ij}}{\eta\sqrt{2\pi}} e^{-\frac{(\lambda_{ij,t})^2}{\eta^2}} \tag{31}$$

where η is the variance of $(\lambda)_{(1..t)}$.

3.3 The final bi-target tracking algorithm

Given all the improvements, we can provide a complete algorithm. See Alg.1 for the algorithm.

Assuming the initial position of the targets are unknown, we will initialize the positions in the sets defined by the original measurements. We then perform the step-by-step estimation.
First of all, we need to perform the decision rule to recognize the situation. After that, we can apply the solutions proposed to estimate the separating straight lines, which

solutions strongly depends on the situation. That leads us to the estimation of the two corrections factor, calculated for each possible association, to perform the best choice possible for each trajectory. Eventually, after the association, comes the final estimation of the positions and the velocity of the two targets.

4 Simulation Results

We tested our algorithm for two targets, moving through a 120mx120m space, during 40 seconds. The number of sensors is 100, (10x10) and are regularly distributed (like a grid). The targets follow the trajectory defined in the first section of the paper. The final assumption we made is that we know the initial sets of each sensors. Then, the initialization of the algorithm is made by a simple shoot on an uniform law. The starting velocity vectors are assumed to be known too.

As we can see in fig. 4, our algorithm tracks the two targets

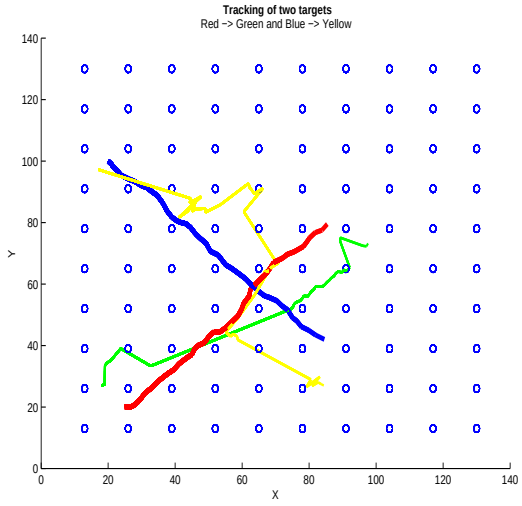

Figure 4: *Bi-Target Tracking with the algorithm.*

pretty well, even if the corrections are not that efficient. After a quite good estimation in the beginning, the algorithm tracks the two targets pretty well until they cross. Then, it seems we have an association trouble, and the two targets seem to be inverted. After a little time, the tracking seems to work again. Despite the crossing, we didn't lose the tracking, and even the simplest behaviour learning seems to be efficient for our tracking issue. This will certainly be outlined in the mean square error calculation through the tracking time. The main difference between the kind of sensors we use and the proximity ones, is that we do not have much trouble is the targets are too close. And if the targets have the same velocity, the association problem is solved by the limitation on the correction λ. Then, the biggest problem comes when the targets are close *and* the velocity vectors are similar. That problem is not handled by the current algorithm.

The next figure (fig. 5) is about the probability of associa-

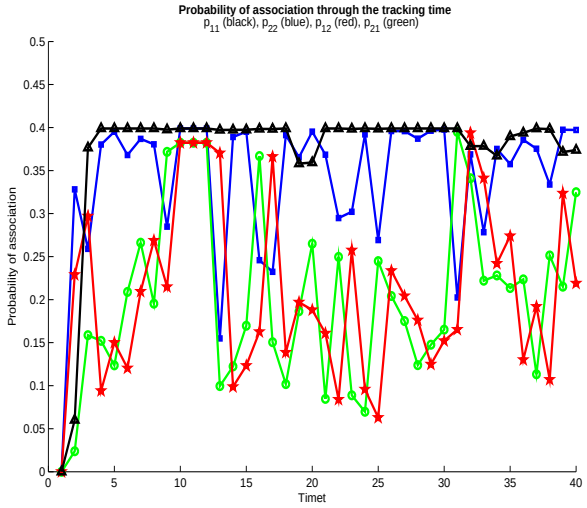

Figure 5: *Probability of association.*

tion, its maximum value, and the differences between two probabilities representing two different associations. The first conclusion we can make is that the maximum isn't that important. It's definitely not a problem, given that we have to choose a scenario. It could be bad if we admitted some false or no detection, in which case a too low probability will leads us to no association and no tracking. This is another problem that has to be solved, but which is not considered yet. Then, what matters is only the biggest probability relatively to the others, and that (most of the time) ends with some very good association decision through the tracking period.

There is also no importance in the label of each probability. Indeed, given an estimated trajectory from the initial time to time $(t-1)$, we estimate two separation lines. Each of this line is *randomly* labeled, before being associated to the most probable trajectory.

More specifically about the results, we can see that very close probability are calculated, and sometimes the decision taken is about very few more probable event. Moreover, we notice a kind of lag of time between p_{12} and p_{21}. It is be very specific to the choosen scenario, given the very similar value of the two velocity vectors.

The last figure (fig. 6) is about the evolution of the mean square error of the position estimation of the two targets through the tracking time. The first simple conclusion is that it seems decreasing. Which is a good new for our algorithm. However, it's not decreasing as fast as it is for a simple target. The reason must be the uncertainty of the association that can sometimes (when the target are crossing) associate target 1 and trajectory 1 until the crossing time, and target 1 and trajectory 2 after it. That event occurs, but quite rarely, which is an explanation on our MSE behavior.

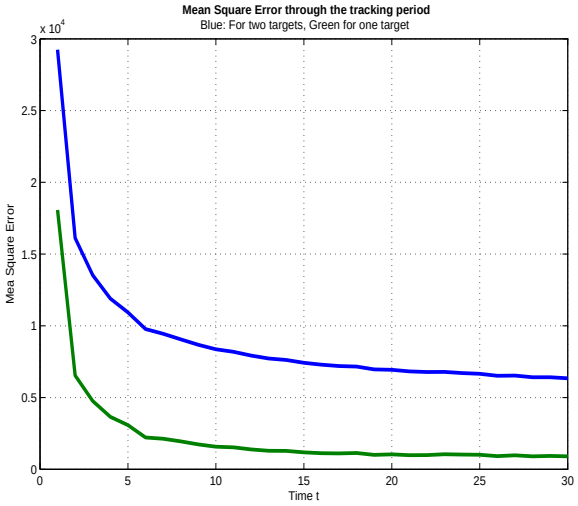

Figure 6: *MSE of the algorithm.*

5 Conclusion

We presented in that paper a very new algorithm to track two targets moving through a space guarded by a sensor network. Each of these sensors have the particularity to give at each time only one bit of information, quite reliable. Despite the poverty of each separated information, we have been able to present a method that dealt with the problem, and provided us a rather good tracking. That algorithm can be divided into four different parts, all of each have the same great importance in the processing, given the dependencies.
The first part can be untitled "classification part", because it has to separate 3 or 4 for instances, given sometimes only 2! We developed some very specific classification methods, completely scenario-adapted, and we finally obtain quite satisfying solutions on that purpose.
The second part is also the less original one, with no new improvement, but is based on classical association method to perform a link from time to time between the past estimated trajectory and the actual estimation of both position and velocity.
The third part is the final tracking part, when we are back to a "classical" one target tracking algorithm given a binary sensor network.
The final part always remain the same, which is perform a backward correction of all the past estimation to increase the velocity estimation (and then, indirectly, the position estimation).
Many improvements can be (and will be) made to increase the quality of the estimation. First of all, instead of considering a very simple "mean-modeling" for the velocity, we could be tempted to perform a more complicated markov model, and estimate the transition probability. Moreover, the association technic remains very simple. There could certainly exist some best-adapted method for the kind of problem we have to face. Especially when the two targets are close, with very similar behaviors. As underline in the simulation results part, we assume that each sensor gives us an information (assumed correct) at each time period. And what if we hadn't the informations about the second target? Finally, we have a quite efficient deterministic tracking algorithm, able to track two targets with quite low errors. The next promising evolution will naturally be a (stochastic?) multitarget tracking algorithm for binary sensors.

References

[1] A. ICKOWICZ, J.-P. LE CADRE,
Target Tracking within a Binary Sensor Network.
Proc. of the 4th International Conference on Intelligent Sensors, Sensor Networks and Information Processing, Dec 2008.

[2] 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.

[3] 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.

[4] C. CORTES, V. VAPNIK,
Support-Vector Networks.
Machine Learning, **20**, 1995, pp. 273–297.

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

[6] J. H. FRIEDMAN and W. STUETZLE,
Projection Pursuit Regression.
J. Amer. Stat. Soc., **76**, 1981, pp. 817–823.

[7] 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.

[8] 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.