# Simulation and Multi-Target Tracking

Edoardo D'Elia, Carlo Tavian, and Alberto Zugno,

**Index Terms**—Camera Simulator, Multiple Target Tracking, SMP, MCMCDA, Data Association, Motion Capture

**Abstract**—Our purpose is to create a camera simulator and a multiple target tracking system for motion capture applications. Starting from literature and state of the art techniques, we focused upon the Markov Chain Monte Carlo (MCMC) method to obtain a MAP estimation of marker's tracks. In particular, this method is based on the well known Metropolis-Hastings algorithm. This method led to very good results (even with false alarms and missing measurements), although it required long computation times. Searching for innovative solutions, we implemented also an algorithm to resolve the Stable Marriage Problem, and we adapted it to the perform data association. This implementation is very fast and results accurate, as drawbacks false alarms and missing measurements are not taken into consideration. For both approaches with provide ideas for possible improvements to get shorter computation times (MCMCDA) or to take into account disturbance factors (SMP).

## 1 Introduction

### 1.1 Goals

#### 1.1.1 Multi-Target Tracking

Multi-target tracking is a well known problem in literature. Given a varying number of targets $k_t$ and measurements $n_t$, the purpose of multi-target tracking is to associate measurements with the targets which have generated them. There are many possible disturbance factors, such as false alarms, corrupted or missing reports, non-constant sampling time, etc.
There are a lot of fields in engineering where multiple-target tracking is widely used, such as motion capture, surveillance systems, computer vision and sensor network. Our interest is focused on motion capture area. In our motion capture application we have to track a variable number of markers that are indiscernible, so the only information we have about them is their position at some sampling time. Given that, our principal problem become the data association: we must distinguish markers resting only upon their dynamics; whereas other applications like computer vision can rely upon some descriptor of tracked objects like color, shape, ecc. that makes data association problem more reasonable. Motion capture is not the only scenario where data association is the principal problem: for example, in a radar tracking of unidentified vehicles (surveillance) we don't have any data other than their position. A few number of algorithms were developed for this purpose; the most widely used are JPDA [1] for a fixed number of target and the MHT [2] for a variable number of targets. These algorithm are used since the Eighties and are generally slow. A new faster method is *Markov Chain Monte Carlo Data Association* [5]. This one outperforms MHT and JPDA in efficiency and accuracy under many conditions. Since a lot of applications in these fields require to be real-time we have implemented the MCMCDA algorithm and we have tested it on some synthetic data. We have also implemented a data association algorithm based on the SMP [3]; this is notably faster than the previous one but as a drawback, it works only with fixed number of targets and without false alarms or undetected measurements (unless we introduce some improvements that we will discuss later).


- *Edoardo D'Elia, ID no. 603406, E-mail: deliaedo@dei.unipd.it*
- *Carlo Tavian, ID no. 607367, E-mail: tavianca@dei.unipd.it*
- *Alberto Zugno, ID no. 607378, E-mail: zugnoalb@dei.unipd.it*

*Students at the Department of Information Engineering, University of Padua. Report presented February 15 2010.*


1. Joint Probabilistic Data Association
2. Multiple Hypothesis Testing
3. Stable Marriage Problem

*Camera Simulator*

In motion capture scenarios a camera simulator should be useful to test various algorithms and to generate synthetic data. In literature there are a bunch of camera models, but only the pin-hole camera model is widely used. We came up with an effective camera simulator which has been used also by other groups which participates in same project.

### 1.2 Results

We managed to implement a MCMCDA algorithm to solve the data association problem. This implementation has turned out quite effective but very slow whereupon not applicable to a real-time application with many targets. We have also adapted the Stable Marriage to Data Association problem and implemented an algorithm that is very effective under some conditions and may be applied in real-time applications.

## 2 Literature and State of the Art

Multiple-target tracking (MTT) is an essential component of surveillance-related systems. Its application areas include missile defense, air traffic control, and — in our case — motion capture techniques. A general formulation of the problem assumes an unknown and variable number of targets that moves with continuity in a given region. In the single-sensor version, the states of these targets are sampled by the sensor and the noisy measurements are provided to the tracking system. The detection probability is less than one then the targets could be undetected at some sampling time. In addition, there are false reports of possible targets or clutter measurements that arise independently from targets of interest. A primary task of an MTT system is data association, i.e. partitioning the measurements into disjoint sets each generated from a single source (target or clutter). Secondary goal the states estimation based on measurements originating from interesting targets. The data association problem could be formulated in several ways. In the single scan data association, the raw measurements are processed one scan at a time and the target states get updated accordingly. Alternatively several sets of measurements could be collected and processed together in batch mode — this is the multi scan data association.

Several methods now exist to handle the data association problem. These may roughly grouped into two categories: Bayesian and non-Bayesian. Among the Bayesian methods, there is the well known Joint Probabilistic Data Association Filter (JPDA) [2], which is a single scan filter where the existent targets' states get updated on the basis of the latest set of measurements (scan). Data association is handled by summing over the probabilities of all feasible partitions, where the targets can't share a measurement and each target could be the source of at most one measurement per scan. Another well known approach is the multiple hypothesis tracker (MHT) [3], in which each hypothesis make a target-to-measurement match and, little by little observations come up, a new set of hypothesis is formulated increasing the number of the previous ones. The hypothesis with

the highest posterior is returned as a solution. MHT is capable of initiating and terminating tracks. However, the number of hypothesis involved in computation grows exponentially over the time. Thus, in order to overcome this computational complexity, a certain number of pruning and clustering methods must be used at expense of optimality.

The non-Bayesian approaches are characterized by hard measurement-to-track association, such that some cost function is minimized. Then the problem may be reformulated as an integer linear programming problem or, to be more precise, as a multidimensional assignment problem, which is NP-hard. Therefore, for the multi scan data association, one should invoke some approximations schemes such as Lagrangian relaxation techniques that relax some constraints and solve the relaxed problem trough linear programming methods [4].

Another option to solve the multi scan data association problem is by using stochastic search methods. In [5] the problem was solved by applying the Markov Chain Monte Carlo (MCMC) method to obtain the partition with maximum posterior. Using the Metropolis-Hastings algorithm, the authors proposed a set of moves to modify the measurements partition, such that sampling from the posterior distribution was possible after few thousands of moves. They showed a remarkable performance of the algorithm in comparison to the MHT method in terms of accuracy of the solution and running time.

## 2.1 Joint Probabilistic Data Association

The Joint Probabilistic Data Association is an extension of the PDA method, which dealt with a single target in clutter, to the situation where there is a *known* number of targets in clutter. When there are several targets in the same neighborhood, measurements from one target can fall in the validation region of a neighboring target (see Fig. 2.1). This can happen over several sampling times and acts as "persistent interference". Since the PDA algorithm models all the incorrect measurements as "random interference", with independent uniform spatial distributions, its performance

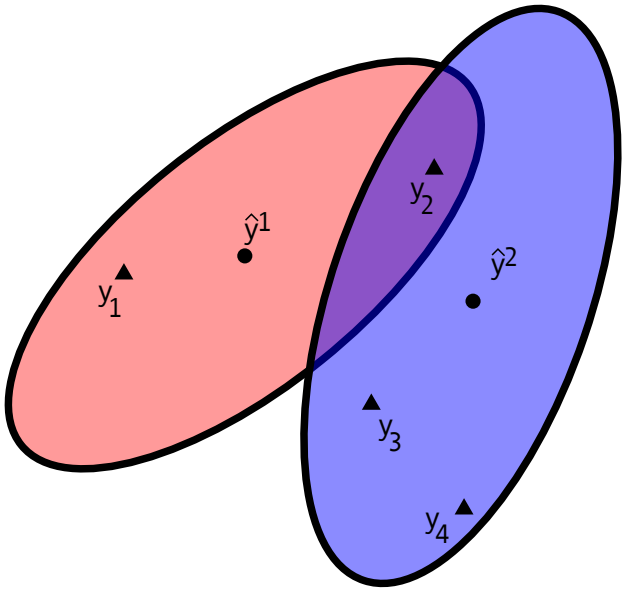



Fig. 2.1. Two targets with a common measurement.

can degrade significantly when the existence of a neighboring target gives rise to interference that is not correctly modeled.

Similarly to the PDA algorithm, the joint probabilistic data association (JPDA) algorithm computes the probabilities of association of only the latest set of measurements $Y_k$ to the various targets — this is a non-backscan approach. Each target has a dynamic and a measurement model. The models for the various targets do not have to be the same. The state estimation is done for each target as in PDA, i.e. a Kalman filtering and a measurement validation, but the measurement-to-target association probabilities are computed in a different way — jointly across the targets.

The key to the JPDA algorithm is the evaluation of the conditional probabilities of the following joint association events pertaining to the current time[4] $k$:

$$\boldsymbol{\theta} = \bigcap_{j=1}^{m_k} \theta_{jt_j},$$

where

$$\theta_{jt} = \{\text{measurement}\, j \,\text{originated from target}\, t\},$$
$$j = 1, \ldots, m_k; \quad t = 0, 1, \ldots, T$$

4. In order to reduce notational complexity, the time index $k$ will be omitted from $\theta, \boldsymbol{\theta}, \omega, \boldsymbol{\Omega}$, etc. whenever it can be done without causing confusion.

and $t_j$ is the index of the target to which measurement $j$ is associated in the event under consideration.

The derivation of the probabilities of joint events is done using Bayes' rule as follows. The conditional probability of the joint association event $\boldsymbol{\theta}(k)$ at time $k$ (with the time argument now indicated explicitly) is

$$\begin{aligned} P[\boldsymbol{\theta}(k)|Y^k] &= P[\boldsymbol{\theta}(k)|Y(k), Y^{k-1}] \\ &= \frac{1}{c} P[Y(k)|\boldsymbol{\theta}(k), Y^{k-1}] P[\boldsymbol{\theta}(k)|Y^{k-1}] \\ &= \frac{1}{c} P[Y(k)|\boldsymbol{\theta}(k), Y^{k-1}] P[\boldsymbol{\theta}(k)] \end{aligned}$$

where $c$ is a normalizing constant. Note that the conditioning on $m_k$ is implicit in the vector $\boldsymbol{\theta}(k)$ and that the irrelevant prior conditioning term has been omitted in the last line of the equation. The PDF on the right-hand side is

$$P[Y(k)|\boldsymbol{\theta}(k), Y^{k-1}] = \prod_{j=1}^{m_k} P[\boldsymbol{y}_j(k)|\theta_{jt_j}(k), Y^{k-1}].$$

The conditional PDF of a measurement given its origin is assumed to be

$$\begin{aligned} &P[y_j(k)|\theta_{jt_j}(k), Y^{k-1}] = \\ &\qquad = \begin{cases} \mathcal{N}_{t_j}(y_j(k)), & \text{if } \tau\{\boldsymbol{\theta}(k)\} = 1 \\ V^{-1}, & \text{if } \tau\{\boldsymbol{\theta}(k)\} = 0, \end{cases} \end{aligned}$$

where a measurement associated with target $t_j$ has the Gaussian PDF

$$\mathcal{N}_{t_j}(y_j(k)) = \mathcal{N}_{t_j}(y_j(k); \hat{y}_j^t(k|k-1), S_j^t(k))$$

and $\hat{y}_j^t$ denotes the predicted measurement for target $t_j$, with associated innovation covariance $S_j^t$. Measurements not associated with any target are assumed uniformly distributed in the surveillance region of volume $V$. In addition the marginal association probabilities are obtained from the joint probabilities by summing over all the joint events in which the marginal event of interest occurs.

## 2.2 Track-Splitting Filter

In this procedure, following initialization, the track is "split" into separate hypothesis tracks at time $k = 1$, one for every measurement at that falls in the validation region around the location $\hat{y}(1|0)$ where the measurement is expected, the track is split. Thus for each measurement a separate updated state and covariance are computed via the Kalman filter equations and propagated forward to yield another validation region at $k = 2$. For each new validation region at $k = 2$, the procedure is then repeated.

Since the number of branches into which the track is split can grow exponentially, the likelihood function of each split track is computed and the unlikely ones are discarded. Denote the $l^{th}$ sequence of measurements up to time $k$ as

$$Y^{k,l} = \{y_{i_{1,l}}(1), \ldots, y_{i_{k,l}}(k)\}$$

where $y_i(j)$ is the $i^{th}$ measurement at time $j$. The likelihood function of this sequence being a track, i.e., of the event that its components originated from the same target, denoted

$$\theta^{k,l} := \{Y^{k,l} \text{is a correct track}\}$$

is the joint probability density function

$$\Lambda(\theta^{k,l}) = P[Y^{k,l}|\theta^{k,l}] = P[y_{i_{1,l}}(1), \ldots, y_{i_{k,l}}(k)|\theta^{k,l}].$$

The track initialization information (initial estimate) is subsumed. The joint PDF can be expressed as

$$\Lambda(\theta^{k,l}) = \prod_{j=1}^{k} P[y_{i_{j,l}}(j)|Y^{j-1}, \theta^{k,l}].$$

Under the linear Gaussian assumptions,

$$\begin{aligned} P[y(j)|Y^{j-1}, \theta^{k,l}] &= \mathcal{N}(y(j); \hat{y}(j|j-1), S(j)) \\ &= \mathcal{N}(e(j); 0, S(j)), \end{aligned}$$

where the last form follows from the definition of the innovations $e(j)$ and the particular form of the Gaussian density. The subscripts on the measurements $y$ have been omitted to avoid making $e$ and $S$ very complicated. From the above equations it follows

$$\begin{aligned} \Lambda(\theta^{k,l}) &= \prod_{j=1}^{k} |2\pi S(j)|^{-\frac{1}{2}} \exp\{-\frac{1}{2} e^T(j) S^{-1}(j) e(j)\} \\ &= c_k \exp\{-\frac{1}{2} e^T(j) S^{-1}(j) e(j)\}. \end{aligned}$$

Note that this likelihood function assumes implicitly that the target detection probability is unity.

The modified log-likelihood function, without explicitly indicating the index $l$, is

$$\lambda(k) = -2\log\left[\frac{\Lambda(\theta^{k,l})}{c_k}\right] = \sum_{j=1}^{k} e^T(j)S^{-1}(j)e(j),$$

which can be also computed recursively as follows:

$$\lambda(k) = \lambda(k-1) + e^T(k)S^{-1}(k)e(k).$$

The last term above has a chi-square density with $n_y$ degrees of freedom and, since the innovations are independent, the log-likelihood function at time $k$ is chi-square distributed with $kn_y$ degrees of freedom. Note that it is a measure of the "goodness of fit" of the measurements.
The statistical test for accepting a track is that the log-likelihood function satisfies

$$\lambda(k) \leq \delta,$$

where the threshold $\delta$ follows from chi-square with $kn_y$ degrees of freedom,

$$P[\chi^2_{kn_y} > \delta] = \alpha$$

In the above, $\alpha$ is the probability that a true track will be rejected and it is taken, typically, as $1\%$.

## 2.3 Multi Hypothesis Tracker

The Multi Hypothesis Tracker (MHT) is a method for calculating the probabilities of various data association hypothesis.
In addition to the above data association capabilities, the algorithm include the desirable characteristic of multiple-scan correlation, clustering and recursiveness. Multiple-scan correlation is the capability to use later measurements to aid in prior correlations (associations) of measurements with targets. This feature is usually found in batch-processing or track-splitting algorithms. Clustering is the process of dividing the whole set of targets and measurements into independent groups (or clusters). Instead of solving a big problem, a number of smaller problems are solved in parallel. Finally, it is desirable for an algorithm to be recursive so that all the previous data do not have to be reprocessed whenever a new data is received.

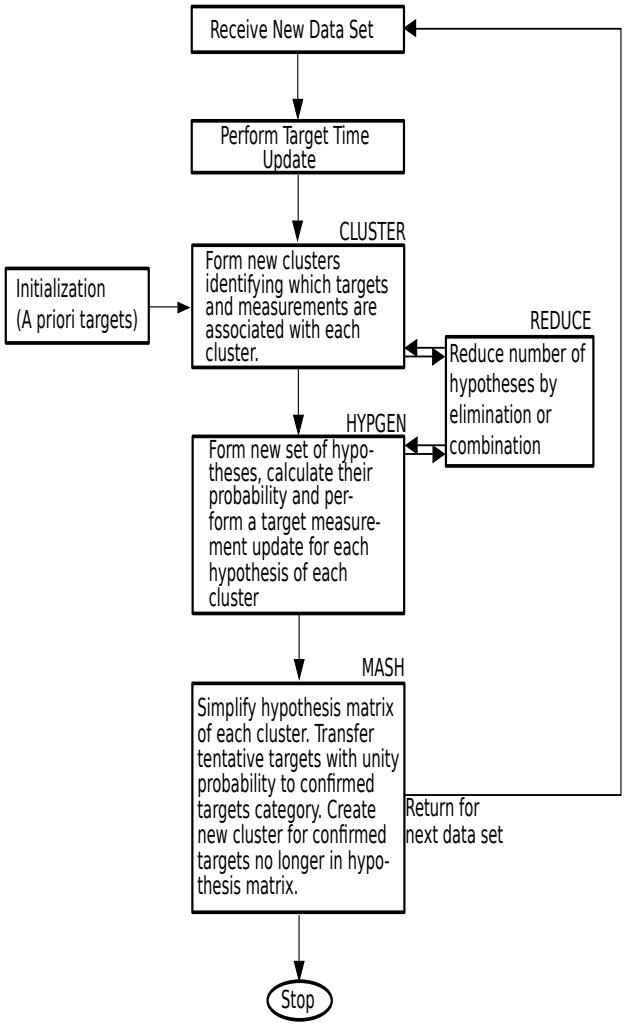



Fig. 2.2. Flow diagram of multiple-target tracking algorithm.

The MHT take into account of some "sensor trouble" by using the detection and false alarm statistics, the expected density of unknown targets, and the accuracy of the target estimates.
A flow diagram of the tracking algorithm is shown in Fig. 2.3. Most of the processing is done within the four subroutines shown in the figure. The CLUSTER subroutine associates measurements with previous clusters. If two or more previously independent clusters are associated because of a measurement, then the two clusters are combined into a "super cluster". A new cluster is formed for any measurement not associated with a prior cluster. As part of initialization program, previously known targets form their own individual clusters.
The HYPGEN subroutine forms new data-association hypothesis for the set of measure-

ments associated with each cluster. The probability is calculated and target estimates are updated for each hypothesis of each cluster.
Both the CLUSTER and HYPGEN subroutines use the REDUCE subroutine for eliminating unlikely hypothesis or combining hypothesis with similar target estimates. Once the set of hypothesis is simplified by this procedure, uniquely associated measurements are eliminated from the hypothesis matrix by the MASH subroutine. Tentative targets become confirmed targets if they were the unique origin of the eliminated measurement.

### 2.4 Linear Assignment Problem

Assignment problems deal with the question how to assign $n$ items (e.g. jobs) to $n$ machines (or workers) in the best possible way. They consist of two components: the assignment as underlying combinatorial structure and an objective function modeling the "best way".
Mathematically an assignment is nothing else than a bijective mapping of a finite set into itself, i.e. a permutation. Assignments can be modeled and visualized in different ways: in a permutation matrix or as an adjacency matrix of a bipartite graph $G = (V, W)$, where the vertex sets $V$ and $W$ have $n$ vertices, i.e. $|V| = |W| = n$.
The set of all assignments of $n$ has $n!$ elements. We can describe this set by the following equations called assignment constraints

$$\begin{aligned}
&\sum_{i=1}^{n} x_{ij} = 1 && \forall j = 1, \ldots, n \\
&\sum_{j=1}^{n} x_{ij} = 1 && \forall i = 1, \ldots, n \\
&x_{ij} \in \{0, 1\} && \forall i, j = 1, \ldots, n.
\end{aligned}$$

Let $A$ be the coefficient matrix of the system of equations. Matrix $A$ is *totally unimodular*, i.e. every square submatrix of $A$ has a determinant of value $+1$, $-1$ or $0$.
Multi-dimensional (sometimes referred as *multi-index*) assignment problems (MAP) are natural extensions of the linear assignment problem. They have been considered for the first time by Pierskalla [1]. The most prominent representatives of this class are axial and planar 3-dimensional assignment problems. The general formulation of the MAP is

$$\begin{aligned}
\min \quad & \sum_{i_1=1}^{n} \cdots \sum_{i_d=1}^{n} c_{i_1 \ldots i_d} x_{i_1 \ldots i_d} \\
\text{s.t.} \quad & \sum_{i_2=1}^{n} \cdots \sum_{i_d=1}^{n} x_{i_1 \ldots i_d} = 1, \quad i_1 = 1, \ldots, n \\
& \sum_{i_1=1}^{n} \sum_{i_3=1}^{n} \cdots \sum_{i_d=1}^{n} x_{i_1 \ldots i_d} = 1, \quad i_2 = 1, \ldots, n \\
& \vdots \\
& \sum_{i_1=1}^{n} \cdots \sum_{i_{d-1}=1}^{n} x_{i_1 \ldots i_d} = 1, \quad i_d = 1, \ldots, n \\
& x_{i_1 \ldots i_d} \in \{0, 1\} \quad \text{for} \quad 1 \le i_1, i_2, \ldots, i_d \le n,
\end{aligned}$$

with $n^d$ coefficient $c_{i_1 \ldots i_d}$.
In terms of graphs a multidimensional assignment problem can be described as follows: let a complete $d$-partite graph $G = (V_1, V_2, \ldots, V_d; E)$ with vertex sets $V_i$, $|V_i| = n$, $i = 1, 2, \ldots, d$, and edge set $E$ be given. A subset $X$ of $V = \bigcup_i^d V_i$ is a clique, if it meets every set $V_i$ in exactly one vertex. A $d$-dimensional assignment is a partition of $V$ into $n$ pairwise disjoint cliques. If $c$ is a real valued cost function defined on the set of cliques of $G = (V_1, V_2, \ldots, V_d; E)$, the $d$-dimensional assignment problem asks for a $d$-dimensional assignment of minimum cost.
Multidimensional assignment problems in their general form have recently found some applications as a means to solve data association problems. More specifically, the central problem in any multi-target tracking and multi-sensor surveillance is the data association problem of partitioning the observations into tracks and false alarms in real time. General classes of these problems can be formulated as multidimensional assignment problems.

### 2.5 MCMCDA

This is the principal method we followed in our work. The algorithm is based on [5] where the authors propose an effective way to solve the MTT problem. The idea is to use a Monte Carlo method to "work out" the formidable Bayesian

estimation matter. To achieve this purpose they used a Metropolis-Hastings algorithm to approximate the optimal solution. More precisely a Markov chain is used for sampling instead of enumerating over all the possible association.
The Monte Carlo Markov Chain Data Association (MCMCDA) algorithm presented in this report is the on-line multiple scan version that includes detection failure, false alarms, and track initiation and termination. In addition, this scheme of filtering is particularly suitable for situation where the number of targets is unknown and changes over the time.
The solution space $\Omega$ contains association histories over multiple steps, as well as considering all possible numbers of targets at each step. The procedure features efficient mechanisms to search over this large solution space in addition to birth and death moves to add or remove tracks.
For further detail see section 4.

### 2.6 Stable Marriage Problem

Imagine you are a matchmaker, with one hundred female clients, and one hundred male clients. Each of the women has given you a complete list of the hundred men, ordered by her preference: her first choice, second choice, and so on. Each of the men has given you a list of the women, ranked similarly. It is your job to arrange one hundred happy marriages.
In this problem, we have a set of n men and n women. Each person has their own preference list of the persons they want to marry. Our job is to determine an assignment where each man is married to one and only one woman (monogamous and heterosexual).
Each man has a list of women ordered by his preference and each woman has a similarly ranked list. Every man is on every woman's list and every woman is on every man's list. The goal is to have a set of stable marriages between the men and the women.
When given two married pairs, (♂, ♀) and (♂, ♀), if man ♂ prefers another woman ♀ more than his current wife ♀ and woman ♀ prefers ♂ more than her current husband ♂, then (♂, ♀) is called a dissatisfied pair. The marriage is said to be a stable marriage if there are no dissatisfied pairs.
Now, our problem is to "marry" targets with measurements in a stable (and satisfied) way. In this report we discuss this method with more detail in section 5.

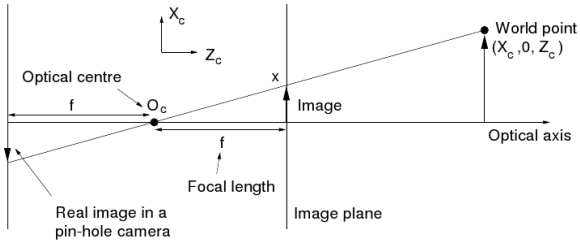



Fig. 3.1. Pin-hole schematics

# 3 CAMERA SIMULATOR

## 3.1 Ideal Camera Model

In order to build a camera simulator we have to define a suitable mathematical model for a generic camera. All camera's aim is to represent 3D points of real world in a 2D surface (the image plane): in order to do this a projection is performed.

The most widely used camera model is the pin-hole camera, which is presented schematically in the figure 3.1. In the pin-hole model we suppose that every ray of light that impress the ccd (or film) and form the camera image has passed through a point called optical center. In a real camera, the fact that all the ray of light pass through the same point is clearly an approximation; nevertheless for small lens this is a very good approximation that can be improved caring of some kind of distortions.

From the knowledge of the focal length and the coordinates $X = [X_c; Y_c; Z_c]^T$ of the world point (relative to a reference frame centered at the optical center with z-axis being the optical axis) we can calculate his projection coordinates ($x = [x, y]^T$) on the image plane.

In a real camera, the image should be upside-down with respect to the object (Figure 3.2) so, for simplicity, we consider the image plane as if it would be located before the optical center. By the property of similar triangles we have that

$$x = \frac{fX_c}{Zc};\ y = \frac{fY_c}{Zc}$$

, where the position of point $(X_c, Y_c, Z_C)$ is referred to the camera's coordinate system. To express the world's point in to camera's coordinate system we must know the position and orientation of the camera (estrinsic parameters) and perform a rototranslation (figure 3.3).

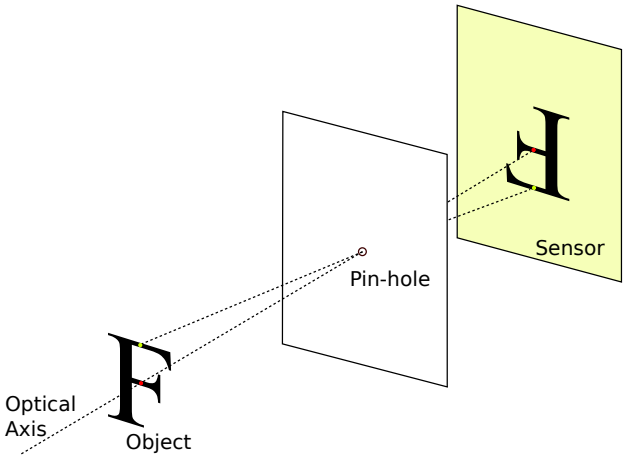



Fig. 3.2. Real pin-hole camera upside-down effect

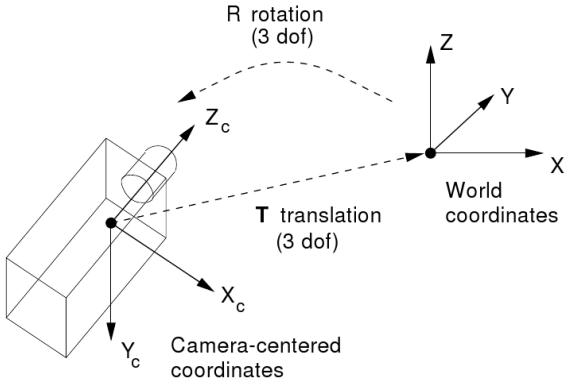



Fig. 3.3. Rototranslation from world to camera frame

Named $R$ the rotation matrix and $T$ the translation vector of the camera-centered coordinate with respect to the world coordinates we have:

$$\begin{bmatrix} X_c \\ Y_c \\ Z_c \end{bmatrix} = R \cdot \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} + T$$

Since all the cameras used today are digital (i.e. the image is represented by a matrix of finite dimensional pixel), it becomes handy to express the position of the projected point using the pixel as measure unit as shown in figure 3.4. Moreover, in a digital image, the origin of the coordinate system is on the top-left corner: $(x, y)$ coordinates are related with $(u, v)$ coordinate by the following equation:

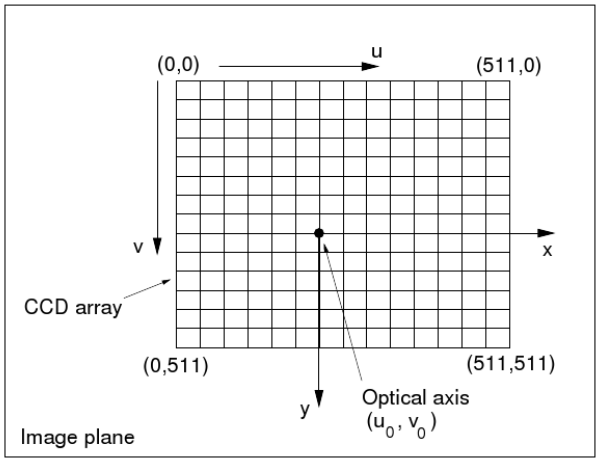



Fig. 3.4. Image reference set translation. Adjust the position of the optical center projection to refer to the image reference set that has the origin in it's upper left corner.

$$u = u_0 + \frac{K_u f X_c}{Z_c}$$
$$v = v_0 + \frac{K_v f Y_c}{Z_c}$$

Where $K_u$, $K_v$ are the pixel width and height, and $u_0$ $v_0$ are the offset of the principal point[5] respect the $(u, v)$ coordinate system ($K_u$, $K_v$, $u_0$, $v_0$ are called intrinsic parameters).
If we express the points position in homogeneous coordinates all these equations for n different points can be rearranged in matrix products:

$$\begin{bmatrix} su_1 & su_2 & \cdots & su_n \\ sv_1 & sv_2 & \cdots & sv_n \\ s & s & \cdots & s \end{bmatrix} = K \cdot F \cdot RT \cdot P_{3dpts} =$$

$$= \begin{bmatrix} k_u & 0 & u_0 \\ 0 & k_v & v_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \cdot$$

$$\cdot \begin{bmatrix} r_{11} & r_{12} & r_{13} & T_x \\ r_{21} & r_{22} & r_{23} & T_y \\ r_{31} & r_{32} & r_{33} & T_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 & x_2 & \cdots & x_n \\ y_1 & y_2 & \cdots & y_n \\ z_1 & z_2 & \cdots & z_n \\ 1 & 1 & \cdots & 1 \end{bmatrix}$$

5. Principal point: the projection of the optical center on the image plane.

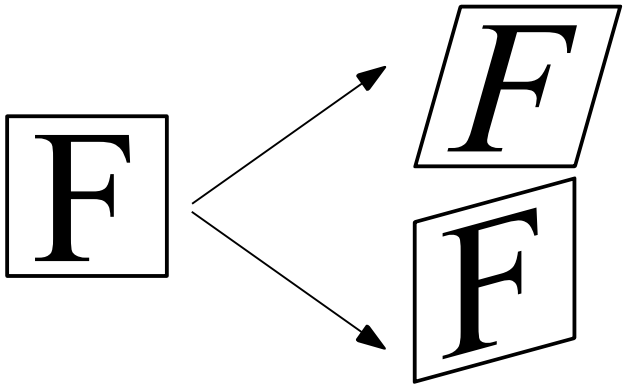



Fig. 3.5. Skew Distortion

Where s is a scale factor. The matrix obtained by the product of first two matrix in the previous equation is called intrinsic parameters matrix.

### 3.2 Distortions

If we want a complete description of the camera we are going to simulate, it could be useful to take into account some kind of distortion.
The simpler distortion we can implement is the skew factor (figure 3.5): introducing the parameter $s_w$ in the intrinsic parameters matrix replacing $K$ matrix with $K'$:

$$K' = \begin{bmatrix} k_u & s_w & u_0 \\ 0 & k_v & v_0 \\ 0 & 0 & 1 \end{bmatrix}$$

Another kind of aberration that typically affects cameras is the radial distortion (Figure3.6). This one is not linear so we can't model it within the previous matrix equations. The simplest effective model for such a distortion is:

$$x = x_d(1 + a_1 r^2 + a_2 r^4)$$
$$y = y_d(1 + a_1 r^2 + a_2 r^4)$$

Depending on the sign of $a_1$ we have two opposite kind of distortion: barrel or pincushion. Barrel distortion [6] (illustrated in figure 3.6) typically will have a positive term for $a_1$ where as pincushion distortion [7] will have a negative

6. In barrel distortion, image magnification decreases with distance from the optical axis. The apparent effect is that of an image which has been mapped around a sphere (or barrel).

7. In pincushion distortion, image magnification increases with the distance from the optical axis. The visible effect is that lines that do not go through the center of the image are bowed inwards, towards the center of the image, like a pincushion.

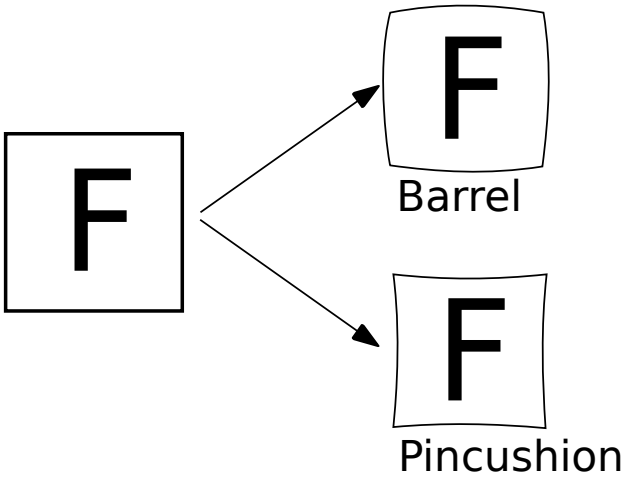



Fig. 3.6. Two type of radial distortions: barrel and pincushion

value. This kind of distortion should be taken into account in the case of pan-tilt-zoom cameras, because appears most visibly when the widest angle (shortest focal length) is selected either with a fixed or a zoom lens.
In addition to radial distortions and skew, there are other types of lens distortion which affect the quality and geometry of an image and are not radially symmetric. This type of distortion (tangential) is not taken into account, since it is usually small compared with that due to radial lens distortion at the typical image resolutions of digital cameras.

### 3.3 Visibility and Visualization

With the previous described model all points are projected on the image plane, but not all are really visible because:

- the projection can be outside the image plane since it has limited dimensions,
- the point is behind image plane
- the point is too far from the camera

The simulator checks, while calculating the position of 3D points respect the camera frame, the distance of any point from the camera and marks as not visible all the points that are behind or too far from it. Similarly, after the projections on an infinite image plane, it marks as not visible all the points that lie outside a finite area of it.
To understand a plot of a complex structure of points could be useful to see the points of different colors and a line structure that links the points. To do this the simulator provides a script to plot the projected points that requires in input a connection matrix that points out which point and which connections we must plot and specify also their colors.

### 3.4 Possible Improvements

In the actual version just geometric points and their connections can be plotted. A useful improvement could be to consider limited polygonal surface, starting with very simple one (triangles and rectangles). The problem in this case is that this surfaces can cover the points behind them and the simulator must mark them as not visible. The simplest algorithm is to project the objects in decreasing order of distance, and mark as not visible old objects as they are covered by the newer. This procedure (Painter's algorithm) is going to fail when there are cyclically overlapping polygons so that is impossible to determine which is the nearest one.
In real applications the most used algorithm is *Z-Buffering* (algorithm 1; used for example in OpenGL and DirectX)

**Algorithm 1** Z-depth algorithm

```
1: for polygon P in the polygon list do
2:     for pixel(x, y) that intersects P do
3:         Calculate z-depth of P at (x, y)
4:         if z-depth < z-buffer[x, y] then
5:             z-buffer(x, y)=z-depth
6:             mark previous (x, y) point as not
   visible
7:         end if
8:     end for
9: end for
10: return
```

The size of this algorithm depends mostly on the resolution of the image because it must calculate the depth of every single pixel and is typically implemented in hardware. There is also a distributed approach called *W-Buffering*. In our case, since we are interested only on the visibility of geometric points (which represent the markers) we could apply this algorithm to the subset of pixel of the camera

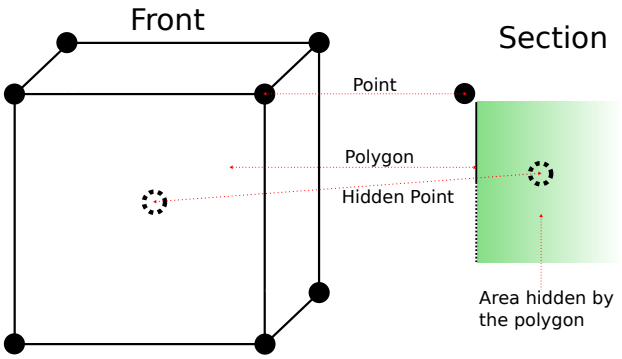



Fig. 3.7. Hidden areas and points. The polygons are not visible on the image plane, but they are used to create some surfaces that hide the points whose projection rays cross their inside.

where a point has been projected. This result in a considerably reduction of complexity, with the drawback that we aren't able to represent correctly the polygon, which can therefor act only as "invisible curtains" that hide the points (figure 3.7).

# 4 MCMCDA

Follow the trajectory of a single particle in clear space is an extreme simple problem. There is no error variance, and no other track to consider. On the opposite side, tracking and labelling multiple objects is generally a NP-Hard problem (as for *JPDA*).
There are some algorithms that solve this problem with a simplified approximated implementation. This reduces the problem to a polynomial computational order. Among various methods, at the state of the art the best one is known as *Markov Chain Monte Carlo Data Association* [7].

## 4.1 Model Formulation

We suppose that there is an hidden Markov chain $\mathcal{M}$ *suitable for modelling track labelling*. Observation time is a sliding window of $T \in \mathbb{Z}^+$ instants, in a region $\mathcal{R}$ with volume $V$. If the current instant is called $H$, the sliding window is the set $\{H-T,\dots,H\}$. Let $K$ be the unknown number of existing or existed target at the present instant $H$. At each time, a target disappears with probability $p_z$ and remains alive with probability $1-p_z$ (a Bernoulli distribution). Otherwise, the number of new arising object at each time is patterned after a Poisson distribution with parameter $\lambda_b V$, where $\lambda_b$ is the birth rate.
The (noisy) Markov chain of the trajectory of each object is, including observing the measurements:

$$\begin{cases} x_{t+1}^q = F^k\left(x_t^q\right) + w_t^q \\ y_t^j = \begin{cases} H^j\left(x_t^k\right) + v_t^j \\ u_t \end{cases} \end{cases} \tag{4.1}$$

where the $j^{\text{th}}$ observation is from target $x_t^k$ and $u_t \sim \text{Unif}\left(\mathcal{R}\right)$ stands for a random process for false alarms. The multi-target tracking and labelling problem is to reconstruct the exact target associations and their trajectories, from measurements. It can be modelled as

$$\begin{cases} \tau_k = \tau_{[H-T,\dots,H]_k} = \text{Alg}\Big(y_{[H-T,\dots,H]}\Big) \\ \tau_{k_a} \cap \tau_{k_b} = \emptyset \qquad \forall a \neq b \in \{0,\dots,K\} \\ \omega = \{\tau_0, \tau_1, \dots, \tau_K\} \end{cases} \tag{4.2}$$

where $\omega$ is the proposed partition of trajectories, taken from $\mathbf{\Omega}$, the space of all possible collections of partitions of any target paths. Obviously, $\text{Alg}(Y)$ symbolizes the selected algorithm in order to compute the most likely tracks from the measurements.
$\tau_0$ is a special array containing all measurements labelled as *false alarms*, invalid data.
The set of all measurements at a certain frame $t$ is

$$Y_t = \left\{ y_t^j \quad | \quad j = 1,\dots,n_t \right\} \tag{4.3}$$

where $n_t$ is the number of observations taken at that istant. The set of all measurements up to the current frame is

$$Y_t = \{ Y_t \quad | \quad t = 1,\dots,H \} \tag{4.4}$$

even though implementations cited in this paper use only a subset, taken from a window of frames:

$$Y_W = \{ Y_t \quad | \quad t = H-T,\dots,H \} \tag{4.5}$$

Finally, given the observations, for a certain distribution $\omega$ there is an *a posteriori* probability $P[\omega|Y]$.

## 4.2 Online Multi-Scan Markov Chain Monte Carlo Data Association algorithm

*Markov Chain Monte Carlo* (aka *MCMC*), as in its Metropolis–Hastings implementation, is a method to generate repeatedly indirect sequences of random samples from an unknown probability distribution, in order to approximate it. Obviously, as assumption is that there are no deterministic ways for extract the exact results.
The standing idea under the *MCMCDA* algorithm is that, at a given time $t$, previous detected tracks $\tau_i^{t-1}$ and measurements up to $Y_j^t$ are part of an hidden (asymptotic) Markov Chain, the cited unknown distribution for the

method. States of this chain are the true targets $x_k^t$ (as well as their trajectory), but all we know are the (noisy) observations $Y_j^t$. The problem is to associate each measurement to the right target, and to recognize the *false alarms*.
MCMCDA is based upon JPDA. In a best case, number of target is known and fixed. Therefore, there is no need to look after all the history of previous detected tracks $\tau_i^{t-1}$ and past measurements. The *single-scan* version si made for that, it is reduced to a simple filtering scheme.
Generally, number of targets is unknown and may change over time. So, it is possible that previous associations may not be correct in light of new evidence: e.g. a track $\tau_i$ might be an union of two tracks, otherwise unused measurements $Y_j^{t-h}$ can produce a trajectory for a new target (appeared at same point in the past).
Unfortunately, the computational complexity of the multi-scan MCMCDA algorithm increases with the time and the number of measurements considered [5]. Otherwise, old associations and observations are less influential then newer [5]. Therefore, it is logical to assume a sliding window of time $[t_{\text{curr}} - t_{\text{win}} + 1, \ldots, t_{\text{curr}}]$, where $t_{\text{curr}}$ is the present instant and $t_{\text{win}}$ the size of the window. Only subsets of $Y$ and $\omega$ are used, therefore generally this version of the algorithm returns a suboptimal solution, but in a reasonable time. This implementation is called *online multi-scan MCMCDA*.
The essence of the algorithm is proposing a random move (chosen following the *proposal distribution* $\omega'$) for a track from past associations and up to current measurements $\omega$. Possible moves are: birth, death, split, merge, extension, reduction, update and switch. If the move is feasible, it is randomly considered as acceptable. Given a number $u \sim \text{Unif}\,[0,1]$, the acceptance pseudo-probability *Acceptance*$(\omega, \omega')$ must be greater than $u$. Typically, *Acceptance*$(\omega, \omega')$ may assume values next to $0$ or is equal to $1$. Finally, the new track distribution is accepted if its posterior conditional probability is greater then the initial's one.
This procedure is iterated $N_{mc}$ times, that stands for the number of iterations in order to reach the asymptotic distribution of the hidden Markov chain. The pseudocode of the algorithm is illustrated in code block 2.

**Algorithm 2** MCMCDA algorithm

**input:** $Y$, noisy 3D points (up to current frame)
**input:** $\omega$, labelling & tracking history of all $\tau_i$ (up to previous frame)
$\hat{\omega} := \omega$

**for** $n = 1, \ldots, N_{mc}$ **do**
    $\omega' =$ *Proposal Distribution*$(\hat{\omega}, Y)$
    $u \sim \text{Unif}\,[0,1]$
    **if** $u <$ *Acceptance*$(\omega, \omega')$ **then**
        $\omega := \omega'$
    **end if**

    **if** $P[\omega|Y] > P[\hat{\omega}|Y]$ **then**
        $\hat{\omega} := \omega$
    **end if**
**end for**

**output:** $\hat{\omega}$

## Proposal Distribution

In order to constructing a Markov Chain with the desired properties, a random move $m$ is chosen for the *Proposal Distribution*$(\hat{\omega}, Y)$, according to a probability distribution $\xi_{K,H}(m)$. If no other condition is given, $\xi_{K,H}(m)$ is modeled as (discrete) uniformly distributed. At the first frame and iteration, when no $\tau_k$ is present, the only possible move is the birth of a track. After the first iteration ($H = 1$) there are only two eventualities: either a track can be deleted either another can originate.
When there is only a single target, no merging or switching between tracks is possible.
If the chosen move is feasible, it is considered as acceptable for a new partition $\omega'$, otherwise the initial $\hat{\omega}$ remains unchanged. Possible moves are: birth, death, split, merge, extension, reduction, track update and track switch. The moves are graphically illustrated in figure 4.1

In [5] and [7] the *Proposal Distribution*$(\cdot, Y)$ has as input $\omega$, not $\hat{\omega}$. This is an error: the proposed $\omega'$ would be computed upon a move that might

pass the acceptance check, but maybe based on trajectories less probable then $\hat{\omega}$ (as $\hat{\omega}$ is the most probable partition). Every iteration, a wronger partition might be extracted and approved for the next cycle. Imposing $\hat{\omega}$ as input to the function, only better partitions can be extracted and approved.
As showed in simulation results section, this fix improves the efficiency of the algorithm from $\approx 15\%$ of target correctly detected and tracked up to $\approx 99.9\%$ of them.

### *Feasible Moves*

**1) Birth Move**. We choose randomly an instant in the sliding window, $t_1 \sim \text{Unif}\{1,\dots,T-1\}$. Then we choose at random a second instant slightly afterwards, $H - t_1 + d_1$, choosing $d_1$ from discrete distribution $d_1 \sim \text{Unif}\{1,\dots,\bar{d}\}$. A birth move is feasible if there are "free" (not already associated to a certain $\tau_k$) data at $H - t_1$ that are, in turn, linkable to other free data at $H - t_1 + d_1$. A link is possible if a path between data at different times can exist, given a maximum velocity-per-frame $\bar{v}$[8].
If the move is feasible, a new track can originate, and the number of all tracks increase: $K_1 := K+1$. Selecting randomly from linkable data at each time $H - t_1$, $H - t_1 + d_1$, $H - t_1 + d_1 + d_2$, etc, the track is recursively extended. Adding measurements is a process that may terminate at a certain frame, with probability $p_z$. The proposed partition is $\omega' = \omega \cup \{\tau_{K_1}\}$[9]
**2) Death Move**. Selected a not-already-deleted track $k_d$ from $\text{Unif}\{1,\dots,K\}$, it will be erased from $\omega$. The partition will be reassigned as $\omega' = \omega \backslash \{\tau_{k_d}\}$.
**3) Split Move**. We select an alive track $k_s$ from $\text{Unif}\{1,\dots,K\}$ with at least 4 instants contained, so $|\tau_{k_s}| \geq 4$. Then, we randomly choose a time $t_r \sim \text{Unif}\{t_2,\dots,t_{|\tau_{k_s}|-2}\}$. A new track originates as $\tau_{k_{s_2}} = \{\tau_{k_{s_2}}(t_{r+1}),\dots,\tau_{k_{s_2}}(t_{|\tau_{k_s}|-2})\}$, and the old one becomes $\tau_{k_{s_1}} = \{\tau_{k_{s_1}}(t_1),\dots,\tau_{k_{s_1}}(t_r)\}$. The global number of tracks increases, $K_1 = K+1$, and the $K_1^{\text{th}}$ is $\tau_{k_{s_2}}$ obviously. The proposed partition will be $\omega' = \big(\omega \backslash \{\tau_{k_s}\}\big) \cup \{\tau_{k_{s_1}}\} \cup \{\tau_{k_{s_2}}\}$.
**4) Merge Move**. For a merge move we must select (uniformly at random) a couple of 2 tracks $(\tau_{k_{s_1}}, \tau_{k_{s_2}})$, the first must end before the latter and they must be *linkable*, as defined above. Then, they will be combined in a single track $\tau_{k_s} = \tau_{k_{s_1}} \cup \tau_{k_{s_2}}$. The proposed partition will be $\omega' = \big(\omega \backslash (\{\tau_{k_{s_1}}\}, \{\tau_{k_{s_2}}\})\big) \cup \{\tau_{k_s}\}$.
**5) Extension Move**. Selected an alive track $k_e$ from $\text{Unif}\{1,\dots,K\}$, we assign free linkable measurements after the last instant of the track $t_{|\tau_{k_e}|}$ as in the birth move. The proposed partition will be $\omega' = \omega \cup \big(\{\tau_{k_s}\} \cup \{\tau_{k_s}(t_{|\tau_{k_e}|+1}),\dots,\tau_{k_s}(t_{|\tau_{k_e}|'})\}\big)$.
**6) Reduction Move**. Select an alive track $k_{rd}$ from $\text{Unif}\{1,\dots,K\}$ and an instant $t_r \sim \text{Unif}\Big\{t_2,\dots,t_{|\tau_{k_{rd}}|-1}\Big\}$. The track will be shortened to $\tau_{k_{rd'}} = \{\tau_{k_{rd}}(t_1),\dots,\tau_{k_{rd}}(t_r)\}$. Measurements after $t_r$ are freed. The proposed partition becomes $\omega' = \big(\omega \backslash \{\tau_{k_{rd}}\}\big) \cup \{\tau_{k_{rd'}}\}$.
**7) Track Update Move**. Selected an alive track $k_u$ from $\text{Unif}\{1,\dots,K\}$, it will be shortened as in the reduction move, from the beginning up to the chosen instant $t_r$. Then, we reassign free linkable measurements after that time. The proposed partition becomes $\omega' = \big(\omega \backslash \{\tau_{k_u}\}\big) \cup \{\tau_{k_{u'}}\}$.
**8) Track Switch Move**. For switching tracks, at first we must (uniformly at random) select a pair of already associated measurements of two distinct tracks in different instants. We name this pair $\big(\tau_{k_{s_1}}(t_p), \tau_{k_{s_2}}(t_q)\big)$. The pair must satisfy the *linking bond*: $\tau_{k_{s_1}}(t_p)$ should be able to reach $\tau_{k_{s_2}}(t_q)$ and its following associations in a time proportional to $\bar{v}$ and $\tau_{k_{s_2}}(t_q)$ should be able to reach the following associations to $\tau_{k_{s_1}}(t_p)$. This is the essence of the switching. Chosen the pair, associations after $t_p$ and $t_q$ are (respectively) exchanged. Switched tracks will become

$$\tau_{k_{1'}} = \Big\{\tau_{k_1}(t_1),\dots,\tau_{k_1}(t_p),\tau_{k_2}(t_{q+1}),\dots,\tau_{k_2}(t_{|\tau_{k_2}|})\Big\}$$

$$\tau_{k_{2'}} = \Big\{\tau_{k_2}(t_1),\dots,\tau_{k_2}(t_q),\tau_{k_1}(t_{p+1}),\dots,\tau_{k_1}(t_{|\tau_{k_1}|})\Big\}$$

Therefore, the proposed partition will become $\omega' = \big(\omega \backslash (\{\tau_{k_1}\}, \{\tau_{k_2}\})\big) \cup \{\tau_{k_{1'}}\} \cup \{\tau_{k_{2'}}\}$.

8. In [?] the set of all linkable data is modelled as mathematical structure, called *the neighborhood tree of measurements*.

9. As implicit assumption, $\omega' = \omega \cup \{\tau_{K_1}\}$ includes the proposing of $\tau_0' = \tau_0 \backslash \{\tau_{K_1}\}$. Equally for other moves.

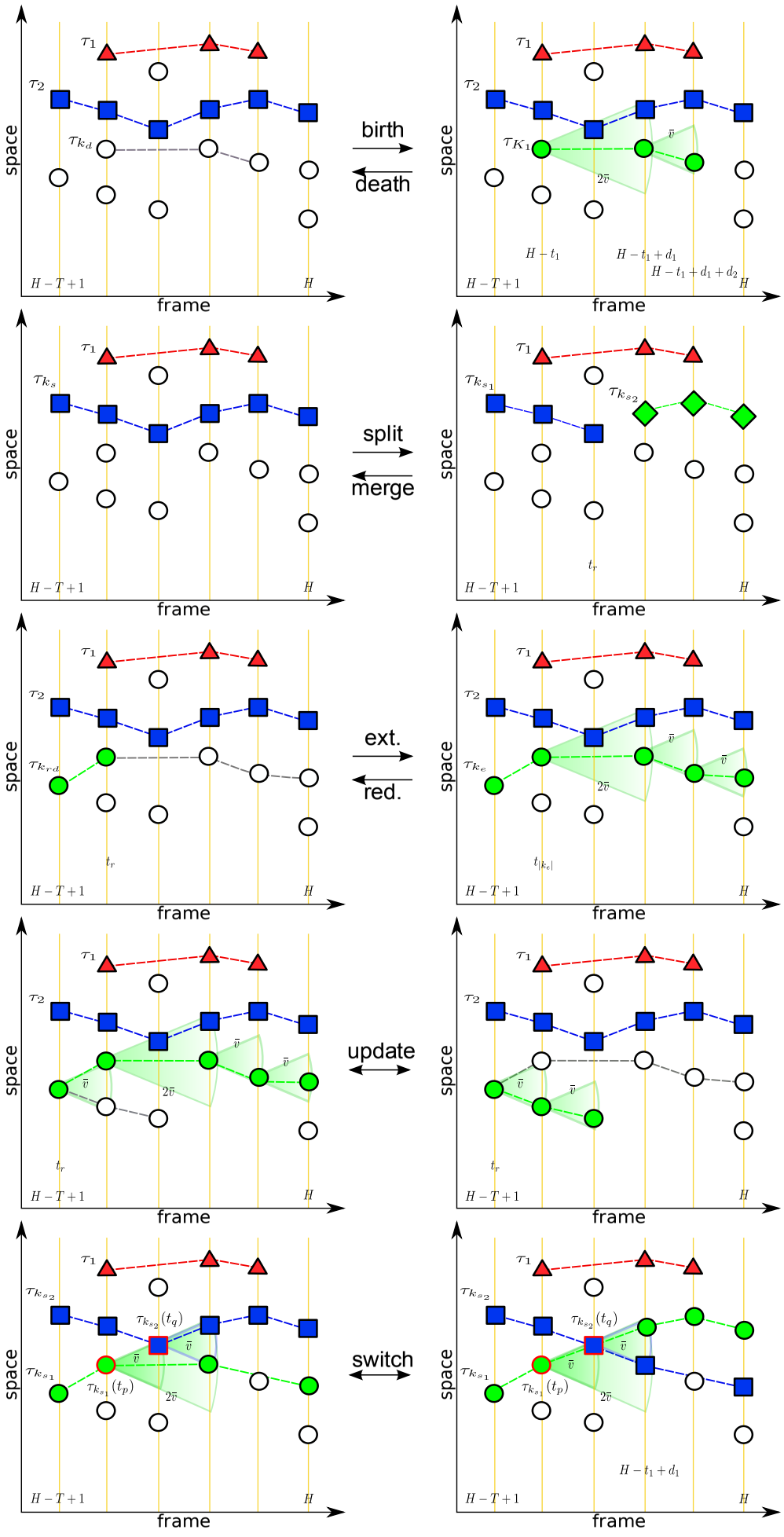



Fig. 4.1. Graphical illustration of feasible moves for a proposal partition.

### Acceptance Probability

We now describe the implementation of the acceptance probability mentioned in the previous sections. The acceptance probability is:

$$A(\omega, \omega') = min\left(1, \frac{\pi(\omega')q(\omega', \omega)}{\pi(\omega)q(\omega, \omega')}\right) \tag{4.6}$$

That is, if we have a feasible solution $\omega$ and we propose a new solution $\omega'$ (chosen at random following the proposal distribution $q(\omega, \omega')$), we accept the move and switch to new solution with probability $A(\omega, \omega')$. In our case the distribution $\pi(\omega)$ is represented by the posterior $P(\omega|Y_{1:t})$ This distribution take into account all the aspect of a feasible solution $\omega$:

$$P(\omega|T_{1:T}) = \frac{1}{Z_0}\prod_{\tau\epsilon\omega\setminus\{\tau_0\}}\prod_{i=2}^{|\tau|}\mathcal{N}(\tau(t_i); \hat{y_{t_i}}(\tau), B_{t_i})\times$$
$$\times\prod_{t=1}^{T} p_z^{z_t}(1-p_z)^{m_{t-1}z_t}\cdot P_d^{d_t}(1-P_d)^{u_t}\lambda_b^{a_t}\lambda_f^{f_t}$$

where $m_t$ denote the number of target at time $t$, $a_t$ the new targets, $z_t$ the terminated ones, $d_t$ the number of detections, $u_t$ the number of undetected targets, $f_t$ the false alarms and $n_t$ the total number of measurements (including false alarms). The value $d_t$ is simply the length of the measurements vector; to determine $z_t$ we have to count how many tracks that at time $t-1$ where associated to a measurements don't have any associations at time $t$; to determine $a_t$ we must count how many new tracks have been created (i.e how many measurements are associated with tracks that have been created at time $t$). Therefore we have:

$$\begin{aligned} m_t &= m_{t-1} + a_t - z_t \\ u_t &= m_t - d_t \\ f_t &= n_t - d_t \end{aligned}$$

Moreover, we have to evaluate the Gaussian $\mathcal{N}(\hat{y_{t_i}}(\tau), B_{t_i})$ at point $\tau(t_i)$ for each track $\tau$ and for each instant $t_i$. This means that, fixed a feasible solution $\omega$, we have to create a Kalman filter for each track starting from the instant where the track begins until the current instant and evaluate the Gaussian at each step. This is computationally the heavier section of the whole algorithm because it must run the Kalman filter on the entire story for each track of each proposed feasible solution $\omega$. Since in equation 4.6 appears only the ratio $\frac{\pi(\omega')}{\pi(\omega)}$ we don't have to compute the normalization constant $Z_0$.

To calculate the value of the proposal distribution, $q(\cdot,\cdot)$, we must take into account the distribution $\xi_{K,H}(\cdot)$ evaluating it at the point $m$ which represents the kind of move we have proposed.

### Possible Improvements

In order to optimizing the MCMCDA algorithm for applying it to marker tracking for motion capture, some enhancements can be used.

#### *Shaping* $\xi_{K,H}(m)$

Motion capture hardware sets are typically optimized to minimize the probability of track's disappearing, $p_z$. For most scenes recorded, number of actors (and therefore markers) normally remains constant for the entire length of the scene (supposing off-screen view as cut from scene shot). So, birth moves after the first frame are not very probable, like death ones. Besides, an extension move is much more likely than the others. To generate a balanced Markov chain, moves such as reduction, track update and track switch have to be almost probable as extension ones. Split and merge ones will be considered less probable.

For example, the distribution for choosing moves after the first frame can be reshaped as

$$\xi_{K,H}(m) = \begin{cases} 0.033 & m=1 & P\,[\text{birth}] = 1/30 \\ 0.066 & m=2 & P\,[\text{death}] = 1/30 \\ 0.133 & m=3 & P\,[\text{split}] = 2/30 \\ 0.2 & m=4 & P\,[\text{merge}] = 2/30 \\ 0.5 & m=5 & P\,[\text{ext.}] = 9/30 \\ 0.666 & m=6 & P\,[\text{red.}] = 5/30 \\ 0.833 & m=7 & P\,[\text{t. switch}] = 5/30 \\ 1 & m=8 & P\,[\text{t. update}] = 5/30 \end{cases} \tag{4.7}$$

At the first frame $\xi_{K,1}(m)$ remains as supposed in a general situation.

### *Two-by-Two Track Distance Bond Pruning*

In motion capture applications, markers are fixed to actor's body, that is a structure with joints. Hence, there are available shape bonds and many measurement are related with each others. An efficient way to capitalize on them is introducing a preliminary check on relative distances. At each frame we compute the distance of each different pair of measurements. This *pre-check* can be easily parallelized. We suppose to use an wide sliding window. Suppose that, when reached the steady state (and therefore a full sliding window is available) we have extracted some tracks with MCMCDA algorithm. If a pair of tracks maintain the same distance from each other during each frame of the sliding window (even rotating), they are considered both as correctly detected target trajectories. The check is made two by two for all tracks. If the distance bond is confirmed for 3 tracks, two by two, their prediction have even less error. With four tracks bonded, a point in 3D space is settled. Only for the current frame $H$, they will be pruned from Metropolis–Hastings method iterations, labeling them as "sure". They can be extended, but not deleted, split, reduced, switched or updated. Tracks that never pass the test can be considered as *free trajectories*, *trajectories with missing measurements* or *false alarms*.

This shape bond speeds up the achievement of the stationary distribution of the Markov chain, as some tracks (states of the chain) have already reached their final state.

A random sequence $U_n$ converges almost surely if $P\big[\lim_{n\to\infty} U_n = U\big] = 1$. Supposing the distance between two tracks at time t in the sliding window as a random variable $d_t =$ distance$(\tau_1, \tau_2)$, a sequence $D_t$ can be modeled as

$$D_t = P\big[d_t = d \quad \forall t \in [H-T+1, T]\big] = \\ = \prod_{t=2}^{T} P\big[d_t = d | d_{t-1} = d\big] \tag{4.8}$$

but, knowing that $d_t = d$ at each frame on a large enough sliding window $[H - T + 1, T]$, we can approximate $P\big[\lim_{n\to\infty} D_n = 1\big] = P\big[\lim_{n\to T} D_n = 1\big] = 1$ that converges with probability 1. Convergence almost surely implies convergence in probability and in distribution, so the MCMCDA algorithm might exclude these tracks from computation.

### *Volume or Distance Clusters Splitting up*

A further refinement to the MCMCDA method might be dividing measurements into groups, assigning to each one a CPU.

As total volume of tracking is known, the original algorithm can be parallelized. Splitting up in smaller portion of space, we can consider as input only a restriction of $\omega$ and $Y$, such as $\omega_v$ and $Y_v$. An instance of MCMCDA runs for each portion of space.

This approach has some limitations. If a region is far from the scene, its parallelized CPU might be unused. Otherwise, some regions have an high density of markers, so their associated CPU are put under much more stress than the others.

A more efficient approach is dividing measurements (and consequently their related track $\tau_k$) into clusters, based on relative distances between 3D data. Set a maximum distance $D_{\max}$, a cluster rises as set of all near measurements in a certain region of space. An instance of MCMCDA runs for each cluster. As said above, computing distances can be easily parallelized. This enhancement is also scalable. Furthermore, any cluster brings to a strongly connected graph, or can be modeled as a small world network with greatest achievable local clustering coefficient (one). This can be capitalized by distributed algorithm.

This approach has some limitations as well. A cluster may be composed by not many measurements, otherwise some other might have a high number of data (such as the "cluster of an hand"). Clusters might also not remain the same during time.

Finally, regrouping tracks (finding the right continuation) of divided measurements remains a common problem for splitting up methods.

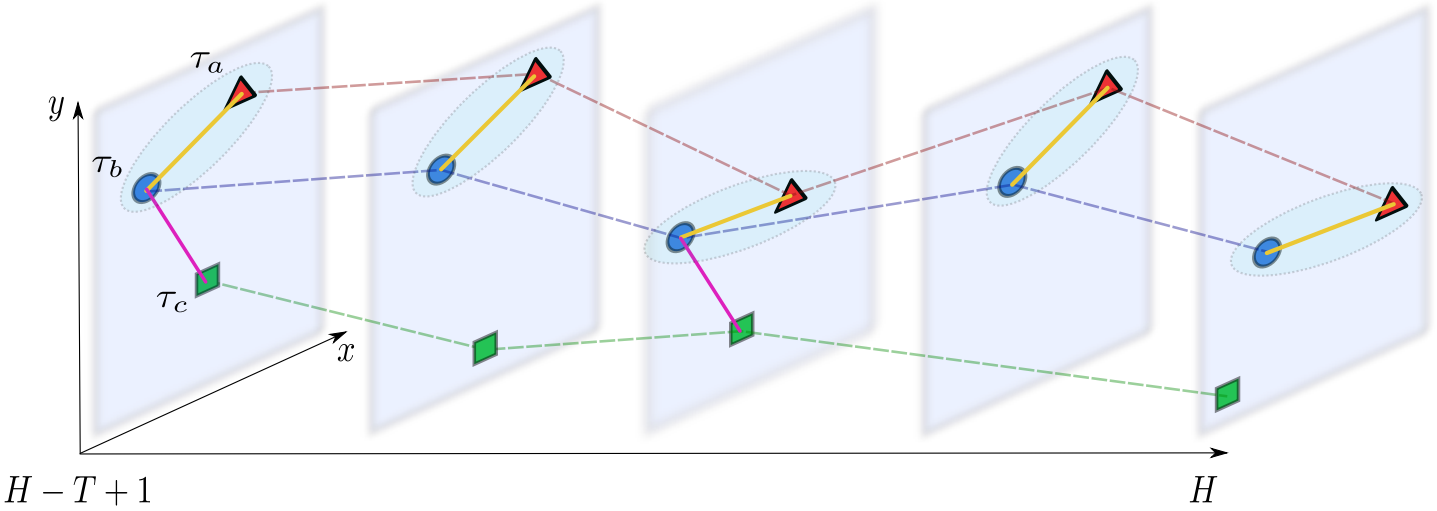
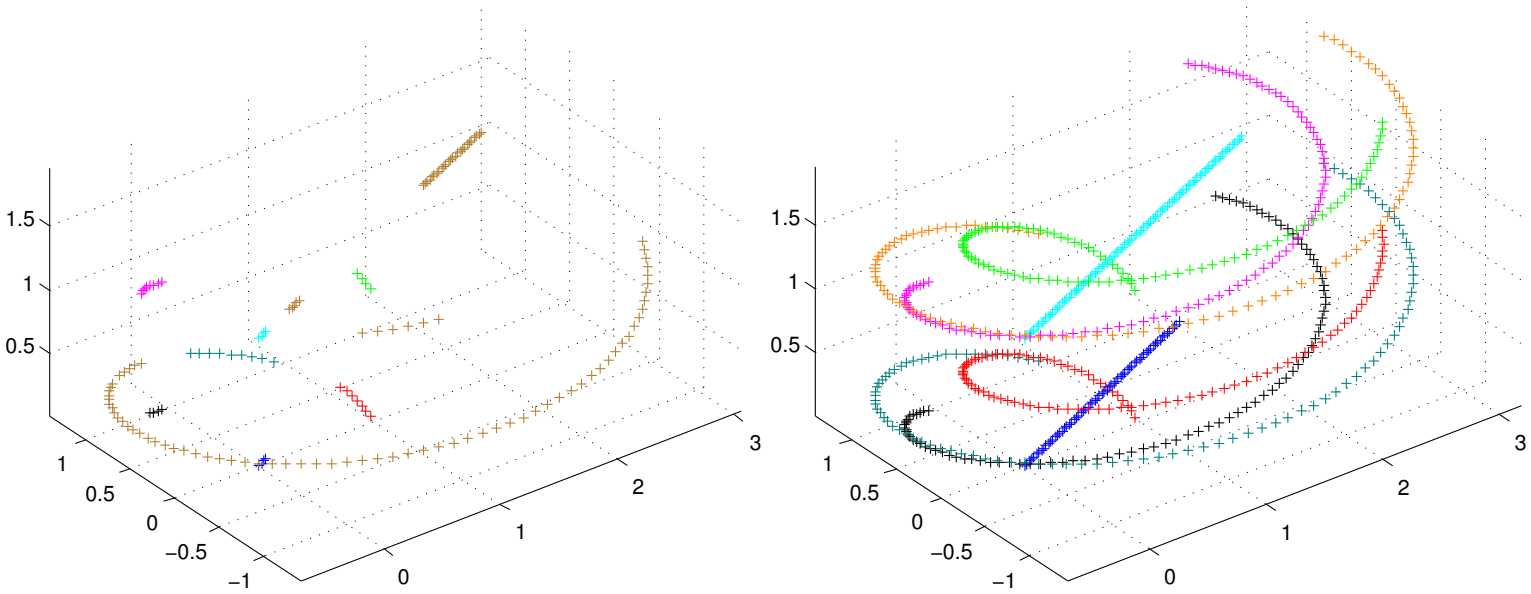



Fig. 4.2. Example of two-by-two track distance bond pruning. Tracks $\tau_a$ and $\tau_b$ respect the bond. Tracks $\tau_a$ and $\tau_c$ are never at the same distance, and $\tau_b$ and $\tau_c$ don't maintain a constant distance for all frames in the sliding window. Besides, $\tau_c$ has a missing measurement. Only $\tau_c$ is not labeled as "sure" in the MCMCDA algorithm.



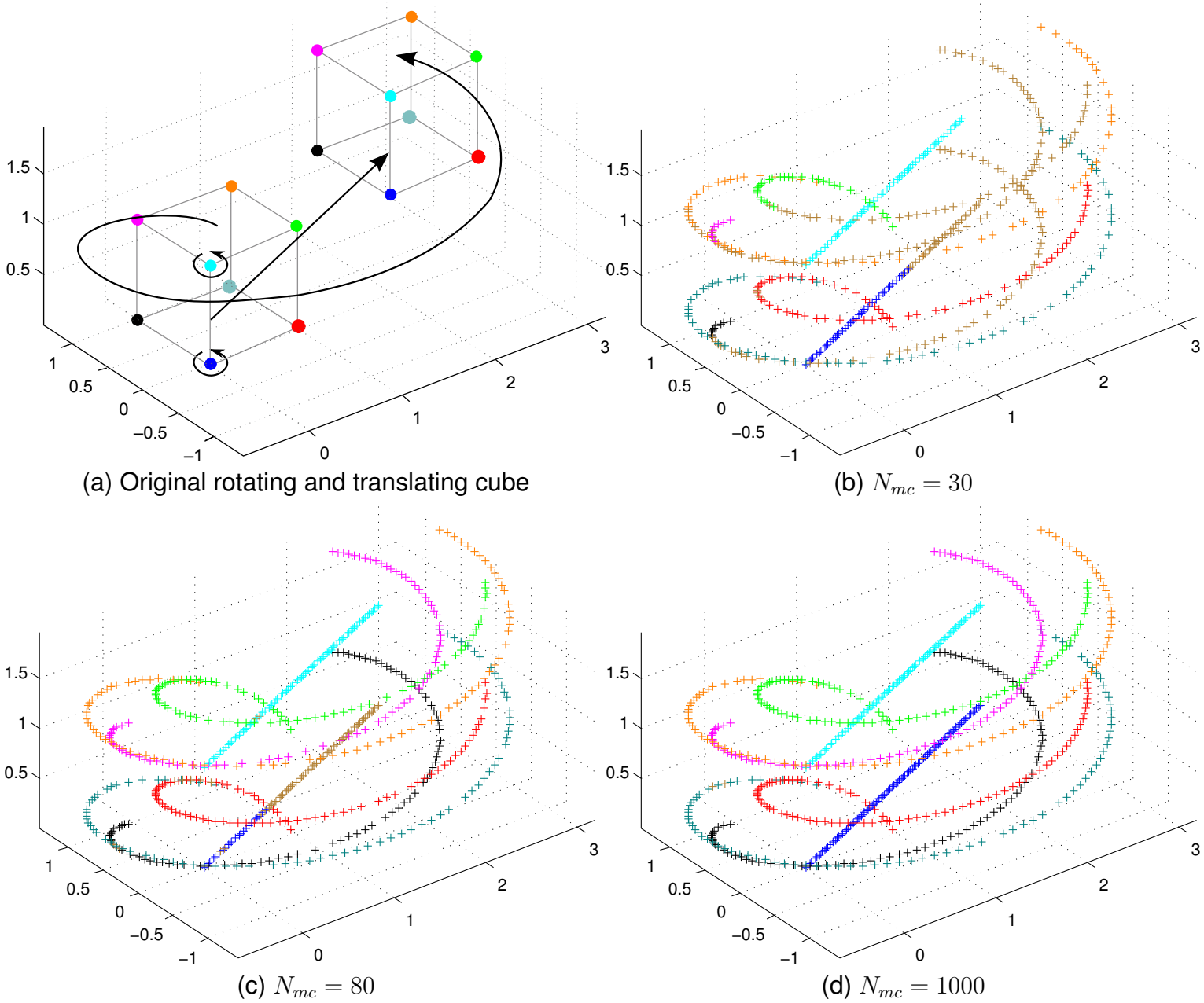
(a) *Proposal Distribution*$(\omega, Y)$ (b) *Proposal Distribution*$(\hat{\omega}, Y)$

Fig. 4.3. Simulation results of a rotating and translating cube (depicted in figure 4.4a), computed with the original algorithm (a) and with the fixed version (b), with $N_{mc} = 30$, no false alarms and no missing data. Brown tracks are labelled as $\tau_k$ with $k > 8$, others are referred to points from $1$ to $8$ of the cube.

### Simulations and results

As shown in 4.3, only $\hat{\omega}$ (as input) is able to get the algorithm to the stationary distribution of the Markov chain, even with few iterations on perfect data.

In Figure 4.3 some simulations computed on the fixed version and improved distribution $\xi_{K,H}(m)$ are illustrated, with optimized MATLAB©code run on a pc with 4 Gb of RAM and Intel Core 2 Duo processor.. As $N_{mc}$ is increasing in order, efficiency of the method can be increased. Also time elapsed raises. If disturbance raises, more iterations have to be done. Missing data don't influence on trajectories only at higher number of cycles-per-frame.



(a) Original rotating and translating cube

(b) $N_{mc} = 30$

(c) $N_{mc} = 80$

(d) $N_{mc} = 1000$

Fig. 4.3. Original moving object and tracking simulation results computed by MCMCDA algorithm, with $10\%$ false alarms and $10\%$ missing data in $100$ frames, increasing $N_{mc}$. Other parameters: $\bar{v} = 0.15$ cm/frame, $\bar{d} = 2$. Brown tracks represent trajectories labeled above $8^{\text{th}}$ target, therefore incorrectly associated.

| Simulation results | | |
|---|---|---|
| $N_{mc} = 30$ | wrong associations | $\sim 36\%$ |
| | generated tracks | 12 |
| | missing labelled marker | $\sim 10\%$ |
| | runtime | 40.70 minutes |
| $N_{mc} = 80$ | wrong associations | $\sim 12\%$ |
| | generated tracks | 9 |
| | missing labelled marker | $\sim 6\%$ |
| | runtime | 102.30 minutes |
| $N_{mc} = 1000$ | wrong associations | $\sim 0.1\%$ |
| | generated tracks | 8 |
| | missing labelled marker | $\sim 0.1\%$ |
| | runtime | 364.75 minutes |

# 5 Stable Marriage Problem

In a data labeling problem we have to associate measurements with entities which have generated them. We thought that the Stable Marriage Problem could became helpful because it provides a way to bind the elements of two different sets in the best way, accordingly to some index that has to be defined.

## 5.1 SMP Definition

Let $M = \{A, B, C, ...\}$ be a set of males and $F = \{a, b, c, ...\}$ a set of females with the same cardinality of $M$. Each male have a ranked list of preference of each female, in descending order. Each female has a similarly ranked list, so that in every female list all man are present and viceversa.
A marriage is a match between one element of $M$ and one of $F$ (e.g. $A - b$). A pair $X - y$ is a dissatisfied one if in the solution $S$ exists two marriages, $X - z$ and $W - y$ such that $X$ prefer $y$ more than is current partner $z$ and woman $y$ prefers $X$ more than her current partner $W$. A set of marriages $M$ is called stable if there are no dissatisfied pair.

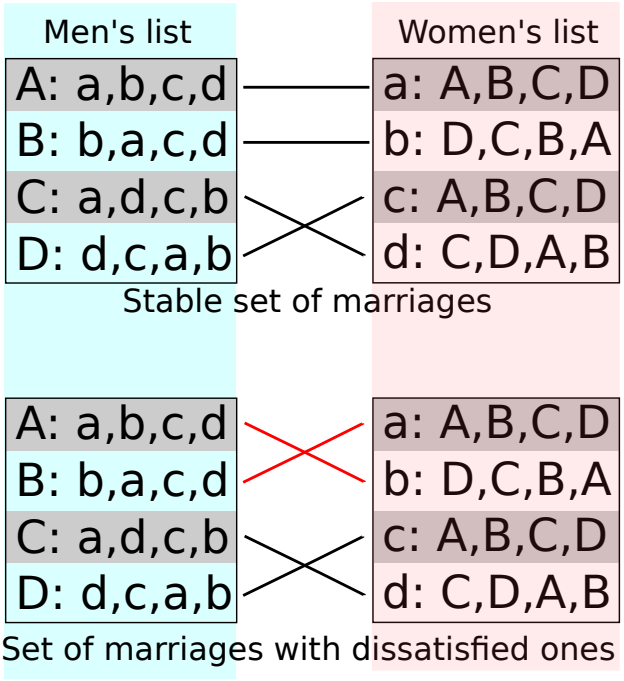



Fig. 5.1. Example of two SMP solutions. The first is a stable set of marriages, the second has some dissatisfied pairs: $A$ prefers $a$ more then is current mate and so do $a$, hence they should be paired in a correct SMP solution.

## 5.2 SMP Algorithm

A simple and deterministic algorithm to determine a stable marriage is shown in algorithm 3

Each man make a marriage proposal to the woman he preferens, and then remove the woman from his list (so that he is not going to make a new proposal to the same woman). Each woman decide to accept the proposal if she is unengaged or if she prefer the new man rather then his actual mate. The algorithm terminates when all males are engaged.

## 5.3 Algorithm Analysis

We have to prove that the algorithm terminates with all men (and women) paired and that

**Algorithm 3** Proposal algorithm

```
while ∃ unpaired man X do
    pick first woman w on X's list
    remove w from X's list
    if w is married then
        if w prefer X to her mate Y then
            set X − w as married
            set Y as unpaired
        else
            X remains unpaired
        end if
    else
        set X − w as married
    end if
end while
return
```

the solution is a stable marriage. Every time that a man make a proposal to a woman, this one is removed from his list. A man whose proposal has been refused will continue making proposals to other women until the end of his list, so that every woman receives at least one proposal. The woman will always accept her first proposal so no woman can be left unmarried. In the worst case the proposal-algorithm makes $O(n^2)$ proposal. To show that it is a stable marriage, let's assume we have a dissatisfied pair, $X - b$, where in the marriage they are paired as $X - a$ and $Y - b$. Since $X$ prefers woman $b$ over his current partner $a$, then he must have proposed to $b$ before $a$. Woman $b$ either rejected him or accepted him, but dropped him for another better man than $X$. Thus, $b$ must prefer $Y$ to $X$, contradicting our assumption that $b$ is dissatisfied, therefore the solution it is a stable marriage set. It can be shown that, although the worst case of the algorithm is $O(n^2)$, the expected (average) case is $O(n \lg n)$ and deviation is small from the expected value (for the complete proof see [8]).

### 5.4 Optimality

We now define a criterion for optimality. A marriage between a man A and a woman B is *feasible* if there exists a stable pairing in which A and B are married. A pairing is *male-optimal* if every man is married with his highest ranked feasible partner. We can prove that the proposal algorithm 3 is a male-optimal one.
Suppose that the solution were not male-optimal. Let be $M$ one man that has just been rejected by his optimal woman $W$. Since the women always accept their first proposal, then $W$ must have been paired whidth a men $Z$ that she prefers more then $M$. $M$ has been just rejected and the proposal are made following the preferences order, that means that $W$ is in the same or higher rank in the $Z$'s preferences list. Moreover, since $W$ is the optimal partner for $M$, there exist a solution $S$ where $M$ is paired with $W$ [10]. But in this solution $S$ we would have that $W$ prefers $Z$ to $M$ and $Z$ prefers $W$ to his mate in $S$. Thus solution $S$ is unstable and this is a contradiction because we have proven that the proposal algorithm provides stable marriages. The solution could be female-optimal if we switch the roles of men and women so that are the women the ones to make the proposal to men.

### 5.5 Application to the Data Labeling

In the data labeling we have two sets (the observations and the targets) we want to put in correspondence. If we make the assumption that:

- the number of marker is costant
- there aren't false alarms
- all the marker are detected

then the number of the targets is equal to the number of observations (measurements) at every time. We can consider the targets set as the males set and the observations set as the females set. We need just to impose a rule to express the preferences both for the males and the females. Let be $M = t_1, t_2, ...t_n$ the set of targets, and $F = y_1, y_2, ...y_n$ the set of new measurements a time T. We calculate the prediction (and variance) for time $T$ for each $t_k$ based on his past history ($[T_0, ..., T]$) with a Kalman filter. Then, fixed $t_k$, we evaluate the Gaussian distribution $\mathcal{N}(\hat{t}_k, \Sigma_{t_k})$ on every $y_i$: sorting these values decreasing we obtain the males (targets) preference list. For the female preferences, we

10. this solution would be achieved if $M$ had made his proposal before $Z$

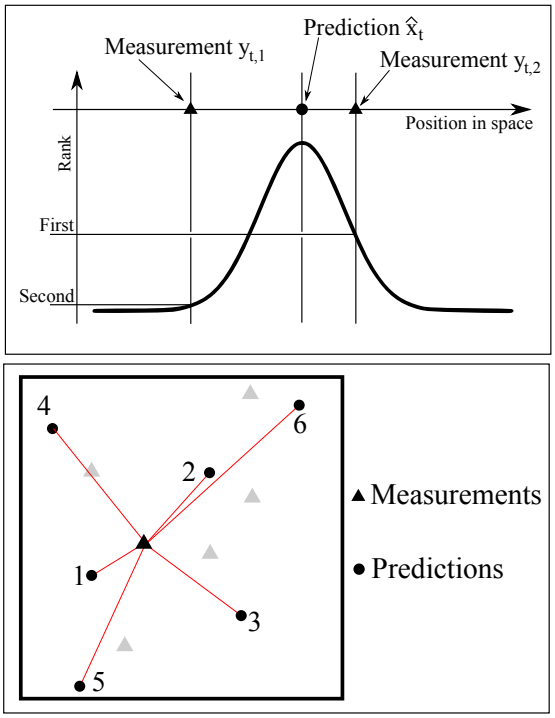



Fig. 5.2. Creating preferences list: males (top) and females (bottom)

simply sort the males by descending order of distance from the female item (figure 5.2).
Since we have proven that the SMP algorithm is male-optimal, we rest assured that, for every single frame, the solution is optimal for the male criterion in the sense of optimality we have previously defined.

## 5.6 Simulation Result

We ran some tests to investigate the behavior of this algorithm under different conditions. The first test we ran was with eight points moving with variable speed and direction in a 3D space. The dynamical model of the system we applied to the Kalman filter was a *Discrete White Noise Acceleration* (DWNA) [11]. Our MATLAB©implementation of SMP took 1.1 seconds to complete the associations over 100 instants; The associations have been always correct. We next run a test with 16 points arranged on the vertexes of two cubes. One of them was still, while the other pass through it while translating and rotating. Also in this case the associations were all correct although at some time the vertexes of the two cubes were lying upon each other. Other more complicated tests show the goodness of this approach. We tried with 8 points following a random walk. Note that in this case would have required to change the Kalman filter used to set the preferences list, since DWNA is not a good model for points that follow a random walk. The associations were all correct even when the points came close to each other. As a test designed specifically to put in trouble our algorithm, we set up 8 points that followed a "particular" random walk, being attracted towards axis origin. Even in this case the associations were very accurate: on the total frames, 82% were correct, 11% frames had one swap between a pair of marker [12] and the rest have more than two wrong associations. See table 5.6 for details.

| #wrong associations | percentage |
|---|---|
| 0 (all correct) | $\sim 82\%$ |
| 2 | $\sim 11\%$ |
| 3 | $\sim 2.5\%$ |
| 4 | $\sim 2.5\%$ |
| 5 | $\sim 0.5\%$ |
| 6 | $\sim 0.5\%$ |
| 7 | $\sim 0.5\%$ |
| 8 | $\sim 0.5\%$ |

.

### Possible Improvements

#### *Variable Number of Targets, False Alarms and Undetection*

The assumptions made on number of targets, false alarms and detection of targets are very restrictive in a real application. It's interesting to analyze solutions that allow to relax these assumptions.
If the number of marker can increase, we simply start some more Kalman filters when we find that the number of observation has increased. Less simple is the opposite case:

11. We choose DWNA because it's a simple model that we think could describe well the marker's dynamic in a wide range of real application in the motion capture scenario. Clearly, to get the best results, the model of the Kalman filter must be adapted to the application.

12. Note that there can't be only one marker mistaken, since if it is wrong, it must have been taken the right place of another one

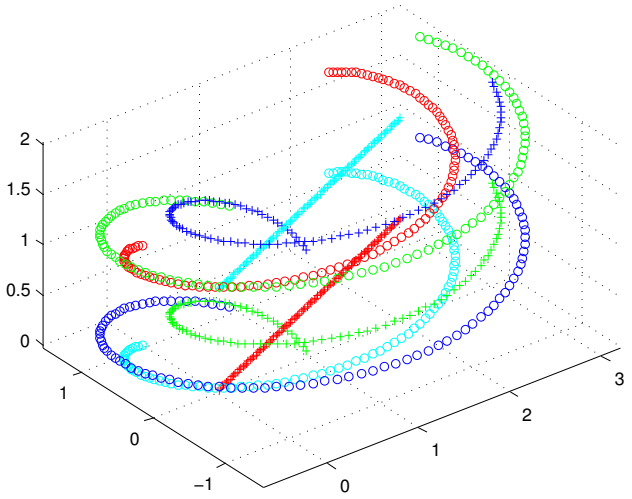


Fig. 5.3. Time evolution of 8 tracked markers lying on the vertexes of a cube tranlating and rotating

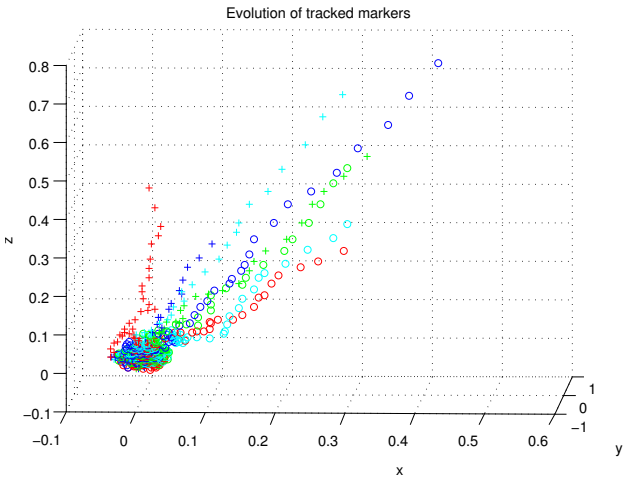



Fig. 5.5. Time evolution of 8 tracked markers attracted towards origin while following a random walk

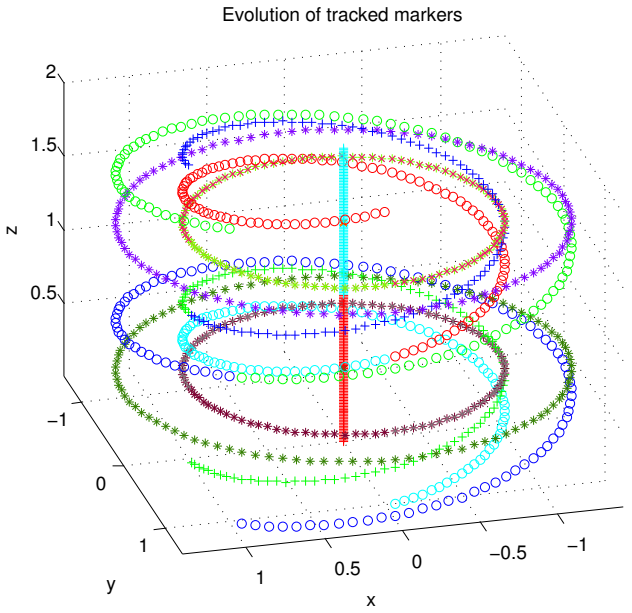



Fig. 5.4. Time evolution of 16 tracked markers lying on the vertexes of a cube tranlating and rotating

if the number of marker can decrease we have to eliminate some tracks, that means to choose which of the previous time target is less likely to have a correspondent measurement at present time. In this case we need to find a rule to delete the less probable measure. One way could be to to analize the females (measures) preference list and keep only the first $K$ [13]

13. with K number of targets (males)

measurements. A work-around that remits the choose of which track to eliminate to the SMP algorithm is to create $n = Nwoman - Nman$ factitious women such that they most certainly are the lower ranked ones for every man [14]. Doing this, we can delete the men (tracks) which are associated by the SMP algorithm to the factitious women.

If we have false alarms or marker that can be undetected, this approach isn't applicable because we have a different number of males and females individual. There are some algorithm that extend the SMP concept to these cases; for example we could adopt the hospitals (or residents) problem. This is variant of SMP where the role of the female individuals is played by the hospitals, that can accept more then one proposal. Clearly, we must provide some other rules to determine which one of the individuals (measures) assigned to the same hospital (target) is the most probable or a way to merge the different measure into a single one.

### *Past History*

For each step our implementation takes into account the whole history for the filtering, but only the last measure to decide the associations

14. For example, we could place them very far away from every target; or we can just fill the males list adding factitious women on the end of the list until we reach the length of $n$ elements.

between the two sets. A way to improve performance could be to redefine the rules for the preferences to take into account all the measure of an interval.

## 6 Conclusions

Our work lead to a MATLAB©implementation of MCMCDA that result too slow for a real-time application. Even if implemented in a more efficient programming language (like C or C++ that are at least one order of magnitude faster) the processing takes too much time. On a quality level, our implementation lead some good results, recognizing correctly the targets in our simulations, that where limited to a few (max 80) instants due to the complexity of the algorithm. The sore point of algorithm is that it has to run a Kalman filter on the whole history (or on the whole sliding window) for each proposed solution and it has to propose a very high number of solutions to have a good approximation of the MCMCDA. To propose a solution applicable to real time application we have adapted the stable marriage problem. Our implementation of this algorithm has results in a fast and effective method for some cases, i.e. where we have constant number of targets and no false alarms or undetected targets. This method have wide leeways to improve quality of associations [15] and, considering it's variants, can be extended to the case of a non constant number of target and with the presence of false alarms and undetected measurements.

15. changing the way individuals choose their preferences

## APPENDIX A
## MARKOV CHAIN MONTE CARLO METHOD

Markov Chain Monte Carlo method is a general method, its aim is extract a probability distribution as stationary distribution of a ad-hoc constructed Markov chain. Normally it is possible to construct a suitable chain for the purpose of the algorithm. This stationary distribution is considered as acquired after a great number of iterations, e.g. $50000$ for approximately $400$ targets [5].
Its applications are, for example, longitudinal studies (like computing the right association between some targets and measurements), numerically calculating multi-dimensional integrals, radiocarbon dating, time trends for desease incidence and mortality [6].
In order to minimize the number of iterations, various random walk algorithms has been used, e.g. *Metropolis–Hastings*, *Gibbs sampling*, *Slice sampling*, *Multiple-try Metropolis*.
John Von Neumann and Stanislaw Ulam elaborated the original idea after 1944, conceived by Enrico Fermi in the 1930s.

## APPENDIX B
## METROPOLIS–HASTINGS ALGORITHM

The Metropolis–Hastings algorithm is an algorithm (used for example by the MCMC method) for obtaining a sequence of random samples from a probability distribution for which direct sampling is difficult. It acts like a random walk at each step, using a proposal density and a method for rejecting proposed moves. The Gibbs sampling algorithm is a special case of the Metropolis–Hastings algorithm which is usually faster and easier to use but is less generally applicable. The algorithm was named after Nicholas Constantine Metropolis.

## REFERENCES


[1] Pierskalla W.P., *The multidimensional assignment problem*. Operations Research 16, 1968, 422–431.
[2] Bar-Shalom Y. and Fortmann T.E., *Tracking and Data Association* Academic Press, San Diego, 1988.
[3] Reid D.B., *An algorithm for tracking multiple targets* IEEE Trans. on Automatic Control, vol. AC-24, no. 6, pp. 843–854, December 1979.
[4] Deb S., Yeddanapudi M., Pattipati K. and Bar-Shalom Y., *A generalized s-d assignment algorithm for multisensor-multitarget state estimation* IEEE Trans. on Aerospace and Electronic Systems, vol. 33, no. 2, pp. 523–538, April 1997
[5] Oh S., Russel S. and Sastry S., *Markov chain monte carlo data association for general multiple-target tracking problems* IEEE Conf. on Decision and Control, 2004.
[6] Spiegelhalter D. J., Richardson S., W. R. Gilks, *Markov chain Monte Carlo in practice* Chapman & Hall 2004.
[7] Oh S., *Bayesian Formulation of Data Association and Markov Chain Monte Carlo Data Association* IEEE Conf. on Decision and Control, 2004.
[8] Hunt W. *The Stable Marriage Problem*
[9] Ma Y., Soatto S., Koseckà J., Sastry S.S. *An Initaion to 3-D Vision* Springer