Titre: Title:	An algorithm for multiple object tracking
Auteur: Author:	Atousa Torabi
Date:	2007
Type:	Mémoire ou thèse / Dissertation or Thesis
Référence: Citation:	Torabi, A. (2007). An algorithm for multiple object tracking [Mémoire de maîtrise, École Polytechnique de Montréal]. PolyPublie. https://publications.polymtl.ca/8115/

UNIVERSITÉ DE MONTRÉAL

AN ALGORITHM FOR MULTIPLE OBJECT TRACKING

ATOUSA TORABI
DÉPARTEMENT DE GÉNIE INFORMATIQUE ET GÉNIE LOGICIEL
ÉCOLE POLYTECHNIQUE DE MONTRÉAL

MÉMOIRE PRÉSENTÉ EN VUE DE L'OBTENTION
DU DIPLÔME DE MAÎTRISE ÈS SCIENCES APPLIQUÉES
(GÉNIE INFORMATIQUE)
DÉCEMBRE 2007

Your file Votre référence
ISBN: 978-0-494-36945-6
Our file Notre référence
ISBN: 978-0-494-36945-6

UNIVERSITÉ DE MONTRÉAL

ÉCOLE POLYTECHNIQUE DE MONTRÉAL

Ce mémoire intitulé :

AN ALGORITHM FOR MULTIPLE OBJECT TRACKING

présenté par : TORABI Atousa

en vue de l'obtention du diplôme de : Maîtrise ès sciences appliquées

a été dûment accepté par le jury d'examen constitué de :

M. LANGLOIS Pierre, Ph.D., président

M. BILODEAU Guillaume-Alexandre, Ph.D., membre et directeur de recherche

Mme. CHERIET Farida, Ph.D., membre

ACKNOWLEDGEMENT

I would like to thank my supervisor, Dr. Guillaume-Alexandre Bilodeau, who has given me all the guidance and counsel I needed throughout this work.

I would also like to thank all my lab mates in LITIV laboratory and the faculty and staff of the Department of Computer Engineering. I am very lucky to have had the chance of being a part of this group.

Special thanks to my family who always give me so much advice, comfort, and love throughout my entire life.

Finally, I would like to thank members of the jury.

RÉSUMÉ

Dans ce travail, un algorithme capable de suivre plusieurs objets est présenté et évalué dans le contexte de l'occlusion d'objet. Les imperfections des méthodes précédentes dans le cas de la fragmentation d'objet et de l'interaction d'objet avec l'environnement ou d'autres objets dans la scène sont également adressées.

L'entrée de notre système de suivi est des Blobs détectées dans la scène. Pour l'étape de prétraitement, notre système a utilisé la soustraction de l'arrière plan qui détecte les Blobs dans chaque trame. Notre algorithme de suivi est basé sur le graph. Dans notre algorithme, l'association de données est exécutée à chaque trame en répétant le processus d'appariement, et en traitant et en mettant à jour un graphe d'événements et un graphe d'hypothèses. Nous avons utilisé l'approche « merge-split » pour traiter des interactions entre les objets. Les changements dramatiques de l'apparence motivent le suivi des objets occlus en exploitant plusieurs trames et en employant l'information globale d'apparence des objets avant de mener l'association de données. Ainsi la représentation en graphe de notre système de suivi, nous permettons d'exploiter l'information dans les trames postérieures et de les lier aux trames précédentes en produisant des hypothèses multiples qui résultent en un processus robuste d'association de données. Notre algorithme est augmenté avec un module qui distingue la division et la fragmentation en surveillant la vitesse des Blobs. Ceci est basé sur le fait que les Blobs réduites en fragments montrent le comportement et le mouvement logiques avec la même vitesse. Pour une séquence de vidéo continue, cet algorithme est augmenté avec un module de correction d'erreurs qui a l'avantage de jeter les hypothèses fausses et de garder seulement les corrects pour optimiser l'exécution. La sortie de notre algorithme est un étiquetage en ligne d'objet et trajectoire également calculée pour chaque objet. Pour établir la trajectoire notre système de suivi rassemble le centre de surface de l'objet, pendant suivre de objet, dans la graphe d'événement et les relie quand l'objet disparait de la scène.

De façon générale, notre système de suivi est capable suivre des objets multiples et manipuler l'interaction d'objets et la fragmentation d'objet. Cet algorithme est conçu

pour être plus efficace en réduisant l'espace de recherche des hypothèses dans la graphe d'hypothèse et plus fiable dans les scénarios compliqués de suivi des objets multiples.

ABSTRACT

In this work, an algorithm for multiple object tracking is presented and evaluated in the context of object occlusion. The shortcomings of previous methods in the case of object fragmentation and object interactions are also addressed.

The input of our tracking system is blobs in the scene. As preprocessing, our system utilized background subtraction that detects the blobs.

We proposed a graph-based tracking algorithm. In our algorithm, data association is performed in each frame by repeating the matching process and updating and processing the event graph and the hypothesis graph. We have used a merge-split approach to handle occlusion (interaction) between the objects. Dramatic changes in the appearance motivate tracking occluded objects by exploiting several frames and using global appearance information of objects before finalizing data association. So the graph representation of our tracking system enable us to exploit the information in later frames and link them to the previous frames by generating multiple hypotheses which results to a robust data association process. Our algorithm is enhanced with a fragmentation checking module which distinguishes splitting and fragmentation by monitoring the velocity of the blobs. This is based on the fact that fragmented blobs show coherent behavior and move with the same velocity. For continuous video stream, this algorithm is enhanced with an error correction module which has the advantage of discarding the wrong hypotheses and keeping only the correct ones to optimize the performance.

The output of our algorithm is object labeling and trajectories for tracked objects. To build the trajectory our tracking system collects the object's centroids during tracking and connects them when object left the scene.

Overall, our tracking system is able to track multiple objects and handle objects interaction and object fragmentation. This algorithm is designed to be more efficient by reducing the search space in hypothesis graph and more reliable in complicated multiple object tracking scenarios.

CONDENSÉ EN FRANÇAIS

Introduction

L'utilisation de caméras pour la vidéosurveillance est commune dans les endroits publics comme les banques, les centres commerciaux, les aéroports, les stations de métro et les sociétés privées. Cependant, les systèmes de vidéosurveillance souffrent d'une lacune importante qui est l'obligation d'une participation humaine active. En effet, un opérateur doit effectuer une surveillance constante de moniteurs. L'efficacité de ces systèmes est donc en grande partie diminuée par l'intervention humaine requise, et non pas par les possibilités technologiques.

Pour surmonter cette limitation, un grand effort de recherche est en cours en vision par ordinateur et en intelligence artificielle pour développer des systèmes automatisés pour la vidéosurveillance en temps réel des personnes, des véhicules, et d'autres objets. Le but final d'un système automatisé de vidéosurveillance est de détecter, reconnaître et suivre dans le champ visuel d'une camera un objet en mouvement, et plus généralement, comprendre et décrire des comportements d'objets aussi automatiquement que possible. Le suivi est une partie majeure d'un système automatisé de vidéosurveillance. Lorsque le système détecte des objets intéressants, le système de suivi doit identifier et trouver les objets dans le champ visuel d'une ou des camera(s). Les deux composantes d'un système de suivi sont:

- la représentation d'objet;
- et l'association des données.

Un objet peut être défini par sa forme, son apparence, ou tout ce qui est intéressant pour l'analyser davantage. Les objets sont identifiés d'une trame à l'autre par un procédé d'association de données.

Dans notre travail, nous sommes intéressés par la mise au point d'un système de suivi approprié pour des applications de vidéosurveillance, capable de suivre plusieurs objets, et qui peut tenir compte des interactions entre les objets comme les fusions et les divisions. Nous visons également un système qui est robuste à certaines fausses détections provenant du module de détection d'objets.

Notre approche de suivi est adaptative. Cela signifie qu'il y a un module de correction qui peut corriger les erreurs de suivi d'objets en explorant l'information provenant de nouvelles trames et en les reliant aux observations précédentes. Notre approche est augmentée par un module de détection de fragmentation qui tient compte de quelques imperfections du module de détection d'objets. Le système proposé est conçu pour:

- traiter les occlusions inter-objets provoquées par les fusions et les divisions des objets dans la scène ;
- avoir une robustesse aux changements d'illumination sur de courts lapses de temps;
- avoir une robustesse aux fragmentations d'objets provenant de la soustraction imparfaite de l'arrière-plan (détection des objets versus l'arrière-plan);
- rassembler pendant le suivi, les informations d'apparence des objets et des événements comme les disparitions, les fusions et les divisions, avec leurs moments d'occurrence (ces informations sont nécessaires pour l'interprétation des comportements des objets et trouver leurs trajectoires) ;
- reconstruire la trajectoire d'objets en reliant tous les centroïdes (centre de masse) des objets pendant le suivi.

Objectifs

Les objectifs de ce mémoire sont:

- Définir une représentation d'objets;
- Développer une méthode de suivi d'objets qui est robuste aux occlusions et aux changements d'apparence;
- Gérer les événements tels que les entrées, les sorties, les fusions et les divisions des objets;
- Valider les algorithmes implantés avec des données témoins.

État de l'art

Un système de suivi est composé de deux parties: représentation d'objets et association de données.

Dans le contexte du suivi visuel, les deux catégories principales pour la représentation d'objets sont: 1) basée sur caractéristiques et 2) basée sur modèle. Pour la catégorie basée sur caractéristiques, les objets peuvent être représentés par de l'information générique telle que les caractéristiques de bas niveau de la forme, de la couleur, des textures, ou par la combinaison de ces caractéristiques. Pour la catégorie basée sur modèle, l'humain et les véhicules peuvent être modélisés par une représentation par squelette, par contours actifs ou par des modèles volumétriques.

Dans le contexte de la vidéosurveillance, les stratégies de suivi sont classées par catégories: 1) approche statistique 2) approche qualitative. L'approche de suivi statistique résout le problème d'association de données par la prise de mesures et en modélisant les incertitudes pendant l'estimation de l'état de l'objet. Cette méthode est appropriée pour le suivi des objets dans une image bruitée. Pour l'approche qualitative, une fonction de coût est définie pour associer des objets dans deux trames ou plus consécutives en utilisant un ensemble de contraintes de mouvement.

Travaux antérieurs pour le suivi d'objet multiple

Le suivi d'objet multiple (SOM) pour des applications de vidéosurveillance, est l'un des problèmes les plus difficiles de la vision par ordinateur. La partie la plus critique d'un système de SOM est l'appariement des objets (association de données) pendant les occlusions provoquées par des interactions entre les objets mobiles ou les structures fixes de la scène surveillée. Dans la dernière décennie, beaucoup de recherches ont été faites pour développer des algorithmes robustes pour des scénarios de vidéosurveillance réalistes. Le sujet de ce mémoire est justement sur les SOM utilisant une camera visible. La détection d'objet est appliquée séparément, et avant le suivi. Dans cette section, quelques définitions élémentaires pour cette catégorie de SOM sont présentées. Elles sont énumérées ci-dessous:

1. *BLOB*: Blob (Binary Large OBject) est une entité primaire, une région mobile indépendante dans l'image qui peut contenir un ou plusieurs objets (groupe d'objets). Les blobs sont décrits par une série d'attributs tels que la position, la vitesse, et l'apparence. Ils sont également caractérisés par des opérations, comme créer (lancer un nouveau suivi), supprimer (enlever un blob), fusionner, et diviser.
2. *Occlusion inter-objets*: L'interaction de deux ou plusieurs blobs où un en cache complètement ou partiellement un autre s'appelle occlusion inter-objet. Dans certaines approches, cette occlusion cause des opérations de fusion et de division des blobs. Elle peut également causer la perte de l'identité d'un blob.
3. *Occlusion due à la structure de la scène* : Ce type d'occlusion se produit quand un blob est caché par un objet stationnaire qui fait partie de la scène (exemple: une colonne). Cela cause, la disparition complète ou partielle du blob pendant un intervalle de temps. Par exemple, une personne qui marche derrière un bâtiment ou derrière un arbre.

4. *Camouflage*: Si un blob est semblable à l'arrière-plan de la scène, alors il peut être impossible de distinguer l'arrière-plan de la scène du blob. Dans cette situation, une région mobile est divisée en plusieurs petites régions qui appartiennent toutes à un même Blob. C'est une limitation des techniques de détection d'objets qui cause ce problème pour le suivi d'objet.

Il y a deux approches principales pour SOM. L'approche qui fait le suivi constamment de chaque objet identifié indépendamment, et l'approche qui fait le suivi d'un groupe d'objets comme un blob pendant les occlusions inter-objets. Dans ce qui suit, ces deux approches sont présentées:

1. Approche merge-split: Dans cette approche, les attributs de chaque blob sont mis à jour sans interruption jusqu'à ce qu'il y ait occlusion. À ce moment-là, le suivi des blobs individuels est gelé et un nouveau groupe de blob est initialisé et celui-ci contient tous les blobs précédents. Le nouveau groupe de blob est suivi et son identité (étiquette) est toutes les identités des objets qu'il contient. La difficulté de cette approche est que quand une division se produit le système de suivi doit identifier l'objet qui se divise du groupe de blob. Les méthodes qui emploient de l'appariement d'objets seulement pour deux trames consécutives (logique séquentielle) peuvent échouer si elles ne peuvent pas identifier les objets dans la trame suivant la division. Mais les algorithmes de suivi qui utilisent les informations de plusieurs trames (logique reportée) pour identifier l'objet après la division, sont plus robustes aux occlusions inter-objets.

2. Approche Straight-through: Dans cette approche, les opérations fusion et division ne sont pas définies. Ceci signifie que les blobs en occlusion ne sont pas fusionnées. Un blob contient toujours au plus un objet. Ainsi dans cette approche seulement les identités simples sont définies et le système de suivi continue de suivre les objets séparément même durant les occlusions. Cette approche doit pouvoir classifier chaque pixel dans la région d'occlusion comme appartenant exactement à un des

objets dans le groupe d'objets en occlusion. La plupart des systèmes se basent sur les caractéristiques d'apparence des objets pour classifier les pixels. Par exemple, dans le travail de [1], des modèles d'aspect avec masques de probabilité sont définis pour des objets et ces modèles sont employés pour identifier les objets pendant les occlusions partielles. Ensuite, la classification de pixels est employée pour identifier la région appartenant à chaque objet dans l'occlusion. La performance de cette approche est directement liée à la qualité des méthodes de segmentation ou de classification de pixels employées. En outre, cette approche échoue dans la plupart des travaux si un objet marche derrière un autre objet mobile et disparaît complètement dans une trame.

Méthode proposée pour le suivi d'objets multiples

Dans ce mémoire, nous présentons un algorithme en ligne de suivi d'objets multiples basé sur l'analyse d'hypothèses multiples pour traiter les incertitudes provenant des occlusions inter-objets.

Dans notre algorithme, les blobs dans le champ visuel de la caméra sont représentés en utilisant l'information du centre de masse du blob, sa taille, son rectangle englobant (c.-à-d. un rectangle autour de la région du blob), son histogramme de couleur, sa vitesse, et son état (c.-à-d. un des événements possibles auxquels l'objet participe, tel que la fusion et la division).

Dans notre algorithme, l'association de données est exécutée à chaque trame en répétant le processus d'appariement, et en traitant et en mettant à jour un graphe d'événements et un graphe d'hypothèses. Nous avons utilisé l'approche *merge-split* pour traiter les interactions entre les objets. Dans notre système de suivi, quand un blob apparaît dans le champ visuel de la caméra, un nœud de suivi est ajouté au graphe d'événements et un nœud d'hypothèse est ajouté au graphe d'hypothèses. Si pendant le suivi, un blob est fusionné (occlusion) avec d'autres blobs, un nœud de suivi est ajouté au graphe d'événements pour le blob fusionné et il est relié avec des arcs à tous les nœuds de blobs participant à l'occlusion. Si une division se produit, pour tous les blobs séparés du groupe, des nœuds de suivi et des nœuds d'hypothèse sont ajoutés au graphe

d'événements et au graphe d'hypothèses respectivement. Pour le graphe d'événements, des arcs relient le nœud de suivi du groupe de blob à tous les nœuds se séparant du groupe. Dans le graphe d'hypothèses, de nouveaux nœuds d'hypothèse sont reliés à tous les nœuds précédents (parents) avec des arcs pondérés (la pondération est la probabilité de chaque paire de nouveaux nœuds reliés). En conclusion, en traitant deux graphes, la meilleure étiquette de l'objet dans chaque trame et la meilleure trajectoire de chaque objet sont trouvées.

Résultats

Notre système de suivi est programmé dans l'environnement *visual C++* et en utilisant la bibliothèque *OpenCV (Open Computer Vision Library)*. L'entrée de notre système de suivi est des trames d'une séquence vidéo dans le spectre visible après l'exécution de la soustraction de l'arrière–plan (détection des objets en mouvement). Le résultat en ligne de notre système de suivi est les étiquettes des objets à chaque trame et le résultat final est la trajectoire de chaque objet suivi dans la séquence vidéo. La soustraction de l'arrière-plan est supposée comme prétraitement pour notre système de suivi. Nous utilisons deux méthodes de soustraction de l'arrière-plan: *RectGauss* (développé dans notre laboratoire (LITIV)) et *Temporal average* [2] Selon la séquence de vidéo, nous avons employé une de ces deux méthodes de soustraction d'arrière-plan comme prétraitement.

Les séquences vidéo que nous avons employées dans nos expériences sont les ensembles de données de CAVIAR et des séquences vidéo prises au laboratoire LITIV. Pour la validation des résultats en ligne des objets identifiés, nous avons employé l'observation qualitative par un opérateur humain; ceci signifie que les étiquettes des objets sont comparées à l'interprétation d'un opérateur humain tandis que le système de suivi s'exécute. Et pour la trajectoire des objets, nous comparons la trajectoire calculée avec les trajectoires-témoins que nous avons produits de façon semi-automatisée. Plusieurs cas de tests ont été choisis pour montrer différents aspects de notre algorithme. Les scénarios de ces cas de tests sont extérieurs, bagage, foule, et fragmentation. Les

résultats montrent que notre méthode est relativement robuste aux imperfections provenant d'erreurs de détection d'objets et aux occlusions, et que les trajectoires obtenues sont en général semblables à celles des trajectoires-témoins.

Conclusion

Nous avons présenté un système de suivi capable de suivre de multiples objets, et qui peut traiter les interactions entre objets et la fragmentation d'objets. Notre approche est adaptative. Cela signifie qu'il a un module de correction en ligne qui peut corriger les erreurs des objets identifiés en explorant l'information des nouvelles trames et en les reliant aux observations précédentes. Notre SOM est également augmenté avec un module pour vérifier la fragmentation qui peut traiter quelques imperfections de la détection d'objets.

Notre algorithme souffre toujours de certains problèmes communs à d'autres systèmes de suivi. La plupart de ces problèmes sont dus à la partie de la détection d'objets (soustraction de l'arrière-plan) et parfois à des suivis incorrects. Pour améliorer les résultats et l'efficacité pour notre système de suivi, nous proposons les correctifs suivants:

- Pour la représentation d'objets, l'ajout de caractéristiques de textures au modèle d'objet peut mener à des modèles plus précis d'un objet, et consécutivement à la génération plus précise d'hypothèses.
- Pour des vidéos très longs, en stockant l'information de la trajectoire des objets dans un fichier et en supprimant les nœuds des graphes qui sont désactivés et qui ne seront jamais réutilisés jusqu'à la fin de la séquence vidéo, on pourrait possiblement obtenir un système de suivi plus efficace.

Travaux futurs

Comme travaux futurs dans le cas du suivi d'humains, pour les étapes de détection d'objets et également du suivi d'objet, nous proposons les améliorations suivantes:

- pour la détection d'objets, la combinaison de la détection de blobs dans une séquence vidéo dans le spectre visible et infrarouge peuvent résulter en une détection plus précise des blobs et consécutivement à l'amélioration du suivi des objets. C'est parce que les résultats de détection d'objets pour les séquences vidéo dans le spectre infrarouge sont plus robustes aux structures de couleurs semblables entre l'arrière-plan et l'objet mobile. D'autre part, les résultats de la détection d'objets pour des séquences dans le spectre visible contiennent de l'information valable sur la couleur et la texture du blob. Ainsi la combinaison de ces deux types d'images peut mener à l'amélioration des résultats de détection d'objets.
- Pour le suivi d'objets, la combinaison des résultats de trajectoires dans les séquences vidéo de spectre visible et infrarouge peut mener à la reconstruction de la partie incorrecte du résultat de trajectoire de la séquence du spectre visible, et ainsi peut mener à l'amélioration des résultats pour la trajectoire.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATION

BLOB	Binary Large OBject
MOT	Multiple Object Tracking
MHT	Multiple Hypothesis Tracking
BS	Background Subtraction
GMT	Ground-truth Marking Tool
FOV	Field Of View
LBP	Local Binary Pattern

INTRODUCTION

Context and problematic

The application of video cameras for monitoring and surveillance is common in both public places like banks, shopping centers, airports, metro stations and private firms. However, the most current video surveillance systems suffer from a major obstacle, which is, the need of human agent involvement for constant monitoring [3]. This also suggests that the effectiveness of these systems is largely affected by the level of human involvement, not the technological capabilities. To overcome this limitation, a major effort is underway in computer vision and artificial intelligence researches to develop automated systems for real-time monitoring of people, vehicles, and other objects [3, 4].

The ultimate goal of an automated visual-surveillance system is to detect, recognize and track objects from video sequences and more generally to understand and describe object behaviors as automatically as possible. In general, an automated visual-surveillance system contains four main modules (Figure 0.1). First in object detection module, the moving objects in the field of view (FOV) of the camera are detected. Second in object classification module, detected objects in the scene are classified such as human or car. Third in object tracking module, detected objects are tracked until they disappear from the FOV of the camera. Fourth in behavior interpretation module, the behavior of each tracked object is analyzed and interpreted as one of the known action defined by system.

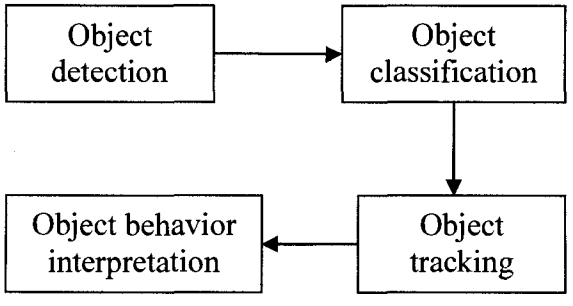

Figure 0.1 automated visual-surveillance modules

The performance of such a system is generally context-dependant and may be compared in a specific application with respect to the quality of decisions, computing time, and robustness to small perturbations [5].

The tracking is a major part of an automated visual-surveillance system. Once the system detects the interesting objects; the tracking system should recognize and find the track of objects in the field of view (FOV) of camera(s). Two main part of a tracking system are: object representation and data association. An object can be defined by its shape, its appearance, or anything that is interesting for further analysis, then the objects should be identified in data association procedure. In a real environment, the tracking system might encounter different difficulties: First, the appearance features of the object, such as shape and size vary over time, especially when we track non-rigid objects like humans. Second, an object can be occluded by a second object; this means the view of one object is blocked by another object. Finally, the object detection methods are not perfect; sometimes they miss the detection of an interesting object or detect uninteresting objects. All these difficulties make tracking a challenging problem in real world scenarios. Researchers have proposed a variety of object modeling and data association algorithms to try to handle these difficulties. Still, they have not been overcome.

In our work, we are interested in an online tracking system appropriate for surveillance applications, able to track multiple objects and handle objects interaction like merging and splitting. We are also looking towards a system which is robust to some specific false detection and miss detection of the object detection module. More specifically, the objectives of this thesis are to:

- Define an object representation;
- Develop an object tracking method which is robust to occlusions and appearance changes;
- Handle events like entering, exiting, merging, splitting;
- Validate of the implemented algorithms with ground truth data.

Overview of the approach

One categorization of object representation is: class-specific models and generic models. The class-specific models require either offline training or designing object models by hand which is not acceptable for real-time surveillance and online applications. The generic models can be applied for online applications. One of the most used generic object representation models for tracking is the color histogram combined with other features like object shape and motion [6-8]. A color histogram is robust to partial occlusions and scale variations and it is fast to build and process. Therefore, its usage is well justified for multiple object tracking. In our work, we used color histogram, velocity, and centroid position to represent an object in the scene.

In multiple objects tracking (MOT), the inter-object occlusion is one of the most common occlusions that happen in a scene. Therefore, data association during and after occlusions is the most critical part of MOT. The graph representation of object tracks is an efficient framework for data association which integrates the spatial and temporal information. This representation enables us to establish patterns and relationship among detected moving regions [9]. Multiple hypothesis tracking (MHT), introduced by *Reid* [10], is one of the most known graph-based MOT methods. This algorithm [10] maintains several correspondence hypotheses for each object in each frame, and the final track of the object is the most likely set of correspondences over the time period of its observation [11]. This approach results in a robust data association because it can recover an incorrect correspondence in a certain frame using deferred decisions. The main drawback of deferred decisions in classic MHT is its costs with respect to memory and processing resources, considering the exponential growth of the decision tree. Moreover, in an extreme case, deferred decisions may lead to batch processing which is unacceptable for a use in an online surveillance system.

Recently, the introduction of MHTs in context of MOT for surveillance applications has resulted in reliable frameworks. In these works, approximations have been proposed to limit the growth of the decision tree by a series of clustering and pruning operations.

In [12], authors formulate the MOT problem in form of a Bayesian network inference problem to find the most probable path for each object in the decision tree. To limit the growth of the tree they have just simply removed dependency links between tracks which may results in a poor data association. In [8], a MOT is enhanced with a HMMs to obtain the best joint probability of the state sequence and the observation sequence. To limit the growth of the decision tree, authors in [8] have used a hypothesis management module to perform local pruning and remove unlikely connections. These works [8, 12] share a common weakness in that the decisions are based only on previous observation. If an error happens on choosing a single state of path based on previous frames information, it will propagate into later frames leading to a poor tracking performance. In [9], a MOT is proposed based on multiple hypotheses about trajectory of objects. They have exploited the information in later frames and linked them to the previous frames which results to a robust tracking system. However, their approach is limited as it requires the whole video before performing data association processing. Therefore, it is not appropriate for online applications.

To the best of our knowledge, no one has proposed an online MOT using the idea of modified MHT proposed in [9]. Moreover, any of the MHT approaches, proposed in the context of surveillance applications, are not considering incorrect object detections caused by the object detection module. In this thesis, we propose a MHT similar to the one proposed in [9]. However, our approach is adaptive. This means it has an online correction module which can correct the errors of objects labeling by exploring the later frames information and relating them to previous observations. Our approach is enhanced with a fragmentation checking module that can handle some imperfections of object detection.

To prepare the input of our tracking system, there is a preprocessing consisting of background subtraction to detect moving region, connected component analysis, shadow elimination to reduce effects of shadows on appearance of moving objects, and BLOB(Binary Large OBject) information extraction (Each independent moving region in the scene which is detected by object detection module, is a BLOB).

Our proposed system is designed to:

- handle inter-object occlusions caused by merging and splitting of objects in the scene;
- have robustness to short time illumination changes;
- have robustness to object region fragmentation caused by the imperfect background subtraction;
- collect during tracking the appearance information of the objects and all object's events like appearing disappearing, merging, and splitting, with their times of occurrence (these information are necessary for object behavior interpretation and finding object trajectory);
- reconstruct the object trajectory by connecting object's centroids of object during its tracking.

Our approach has three main advantages over the previous MHT systems:

- Previous MHT in [8, 12] has sequential approach to generate hypothesis, this means hypotheses are generated only from the related parents one level upper than leaf node. The limitation of this approach is that they compute a lot of redundant paths so they need approximations to reduce size of decision tree. Our approach generates hypotheses from all the related parents in upper levels of graph to leaf node and does not compute and store all possible paths. The advantage of this approach is that it does not need approximations and does not propagate error of data association in one frame to later frames. Instead, it recovers errors happening in a frame by relating information of previous frames to later frames in hypothesis graph.
- Our system has a fragmentation control module which differentiates between splitting caused by separation of independent moving regions and fragmentation caused by imperfect background subtraction. This module helps initializing hypotheses node only for real splitting. This results in a reduced size for the decision tree and an accurate data association.

- Our approach uses adaptive color histogram to generate hypotheses. Adaptive color histogram contains global color information about the objects during several frames. This results in a more precise hypothesis generation.

The remainder of this thesis is organized as follows. In chapter 1 we present an overview of object representation methods and a survey of object tracking techniques; we also present a brief introduction about occlusion handling approaches and a short background about concepts that we used in our method. In chapter 2, we describe our tracking method in details. In chapter 3 we present our results and discussion. Finally in chapter 4, we conclude and discuss the future directions in this area of research.

CHAPTER 1 STATE OF THE ART

Tracking is an important task of computer vision with many applications in surveillance, object motion and behavior analysis, navigation, sport scene analysis, and video database management [5]. A tracking system contains two parts: object representation and data association (Figure 1.2).

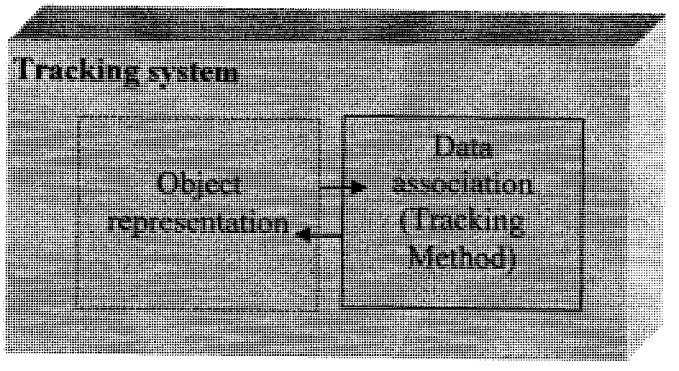

Figure 1.1 Tracking system

These two parts have considerable overlap; this means that the object representation has a direct effect on the results of the tracking method and conversely the tracking method should be designed appropriately for the object representation.

This chapter is organized as follows, in section1.1, background for multiple object tracking is described, in the section 1.2, the object representation methods are presented, and in section 1.3, the tracking strategies are explained.

1.1 Background for Multiple object tracking

Multiple object tracking (MOT) for surveillance applications, is one of the most challenging problems of computer vision. The most critical part of a MOT system is matching objects (data association) during occlusions caused by interactions between moving objects or structures of the monitored scene. In the last decade, many researches have been done for developing robust algorithms for realistic monitoring scenarios. The focus of this thesis is on MOT based on a single visible camera. Object detection is applied separately and before tracking. In this section, some elementary definitions for this category of MOT are presented as follow:

1. *Blob*: Blob (Binary Large OBject) is a primary entity; an independent connected moving region in the image which may contains one or many objects (group of objects). Blobs are described by series of attributes such as position, velocity, and appearance. They are also characterized by operations, such as create (initiate a new track), delete (remove a blob from future consideration), merge, and split.

2. *Inter-object occlusion*: Interaction of two or more blobs where one hides completely or partially another is called inter-object occlusion (Figure 1.2). In some approach, this occlusion caused merging and splitting operations for blobs. It may also cause lost identity/identities of a blob.

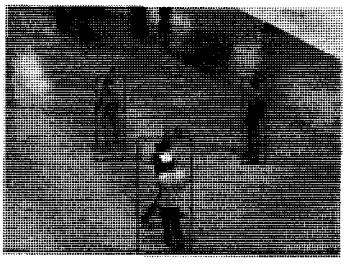

Figure1.2 Two interacting people.

3. *Occlusion of objects due to scene structure*: This type of occlusion happens when a blob interacts with a stationary object which is part of the scene

(Figure 1.3). This cause, some part of the blob or the complete blob disappears for certain period of time. For example in Fig. 1.3, a person is walking behind a desk.

Figure 1.3 A person hidden behind a desk (source[13]).

4. *Camouflage*: If a blob is similar to the background, then it might not be possible to distinguish between the background scene and the blob (Figure 1.4). In this situation a moving region is split to smaller regions which are all belonging to one blob. This is a limitation of object detection techniques but it cause problem for object tracking.

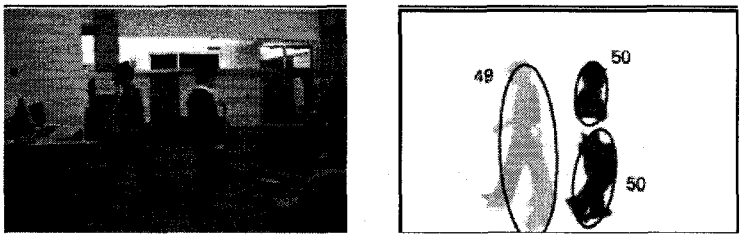

Figure 1.4 A video frame before (left image) and after (right image) background subtraction (source: [14]).

There are two main approaches for MOT: 1) merge-split 2) straight- through. In the following, these two approaches are presented in detail:

1. *Merge-split approach*: In this approach the attributes of each blob are continuously updated until they come into occlusion. At that time, the tracking of individual blobs is frozen and a new group blob is initiated which contains all previous blobs. The new group blob is tracked and its identity (label) is all the

identities of the objects in it. Also, its attributes are continuously updated as other blobs. The difficulty of this approach is when a split condition occurs and the tracking system needs to identify the object that is splitting from the group blob. The methods which use from two consecutive frames object matching (sequential logic) may fails if they cannot identify objects in the exact frame after splitting. But the tracking algorithms which use from several frame information (deferred logic) to identify the object after splitting are more robust to inter-object occlusion.

2. *Straight-through approach*: In this approach, the merge and split operations are not defined. This means occluding blobs are not merged. A blob always contains at most one object. So in this approach only the single identities are defined and the tracking system continues tracking individual objects even through occlusion. This approach needs to be able to classify each pixel in the occlusion region as belonging to exactly one of the occluded objects. Most systems rely on the appearance features of objects to classify these pixels. For example, in the work of [1], probability mask appearance models are defined for objects and these models are used to localized objects during partial occlusion. Then the pixel classification is used to identify the region belonging to each occluding/occluded object. In another work [15], the color histogram and correlogram is used for color modeling of objects and a segmentation method is used to identify individual objects during occlusion. The performance of this approach is directly related to quality of segmentation or pixel classification methods that people used in their work. Furthermore, this approach fails in most works if an object walks behind another moving object and completely disappears in a frame.

1.2 Object representation

Object could be an independently moving person, vehicle, or animal in the field of view (FOV) of camera(s). For tracking purposes, objects should be defined as all the interesting information for further analysis. In the context of visual tracking, the two main categories of object representation are known as: 1) feature-based 2) model-based (Figure 1.5).

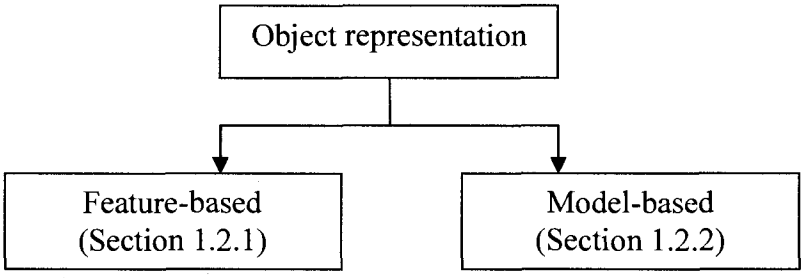

Figure 1.5 Object representation categories

1.2.1 Feature-based representation

Objects can be represented by generic information such as low-level features of shape, color, texture, or by the combination of these features. The features can be extracted from global or local information of regions belonging to the moving object in the image. The local features are usually line segments, points, corner vertices, and edges and the global features are usually color, area and texture. The four main categories of visual features are shape, color, texture, motion. They are described in the followings.

1.2.1.1 Shape

In the feature-based representation, the low-level information of shape such as points, lines, area, and edges, are used. In many of the previous works [8, 16, 17], shape features are combined with other appearance features like color and texture to represent an object.

You can see the state-of-the-art methods for shape descriptors (Region descriptors) that is widely used for visual tracking in follow.

One way to describe the shape is the geometric moments of the object in binary image. It is defined as

$$u_{pq} = \sum_{i=1}^{N} \sum_{j=1}^{N} i^p j^q f(i,j). \tag{1.1}$$

In Eq. 1.1, N is the number of object's pixels, (i,j) is the coordinates of object's pixel, and $p+q$ is the order of the moment. The object's pixels in binary image have $f(i,j) = 1$.

In context of visual tracking, area is one of the features to represent the object. Area is the order zero of geometric moments. In other words, area is the number of pixels of an object or the size of an object. The size is a useful feature during tracking because objects can be distinguished by their size. In a video sequence, the size of the tracking object varies gradually over the time. When an object moves towards the camera, the object gradually becomes bigger and when the object goes farther from the camera, the object gradually become a smaller. In addition, in the frame where an object becomes occluded, the size of the object changes a lot. This implies that the size can be used as a feature to identify occlusions. Area is defined as

$$Area \rightarrow u_{00} = \sum_{i=1}^{N} \sum_{j=1}^{N} i^0 j^0 f(i,j). \tag{1.2}$$

Another feature extracted from the zero and the first order moments are centroid or mean. The centroid is meant to represent the center of mass of an object region. In the tracking context, the trajectory of an object is defined as the curve passing thru the positions of the centroid of the object regions in several frames. Centroid is defined as

$$Centroid \rightarrow X_c = \frac{u_{10}}{u_{00}}, Y_c = \frac{u_{01}}{u_{00}}, \tag{1.3}$$

Where X_c and Y_c are the position of the centroid for the X and Y axes of the image.

The variance is a shape feature extracted from the second order moment about the centroid. The variance describes how symmetric (circular) is the shape. It is defined as

$$\sigma_x^2 = \frac{\sum_{i=1}^{N}\sum_{j=1}^{M}(i - X_C)f(i,j)}{u_{00}} \text{ and} \tag{1.4}$$

$$\sigma_y^2 = \frac{\sum_{i=1}^{N}\sum_{j=1}^{M}(j - Y_C)f(i,j)}{u_{00}}. \tag{1.5}$$

Eq. 1.4 is the horizontal variance and Eq. 1.5 is the vertical variance. Based on the shape moments, a set of moments partly invariant to rotation, translation, and scale variation are used for object representation in image retrieval and object tracking [18, 19].

Another way to model a region is using projections. For example, W4 [16] is a tracking system which uses extensively this approach to track people and find events like carrying objects. In this work, the major axis of the blob, which goes through the middle of an object region, is detected by performing PCA (Principal Component Analysis). Then horizontal and vertical projection histograms of the blob are computed along and perpendicular to the major axis. Using the fact that human shape is symmetric around its major axis, carrying object is detected by extracting non-symmetric regions in the person's silhouette. Moreover; by analyzing projection histograms, periodic movement

(i.e. like walking) of a human is recognized; so, a person is distinguished from other moving objects.

1.2.1.2 Color

Color information is the most used region-based feature for object representation in the context of object tracking. Each pixel in a digital image can be represented as a point in a three dimensional color space. There are many color spaces such as YCrCb, CIELAB, CIE XYZ, CMYK, YBR, HSV, HSL, and RGB. Each of these color spaces is appropriate for a specific application. The RGB and HSV are two common color spaces in image retrieval and tracking and they can be easily converted from one another by simple formulas [20]. In figure 1.6 below, the RGB and HSV color space are shown.

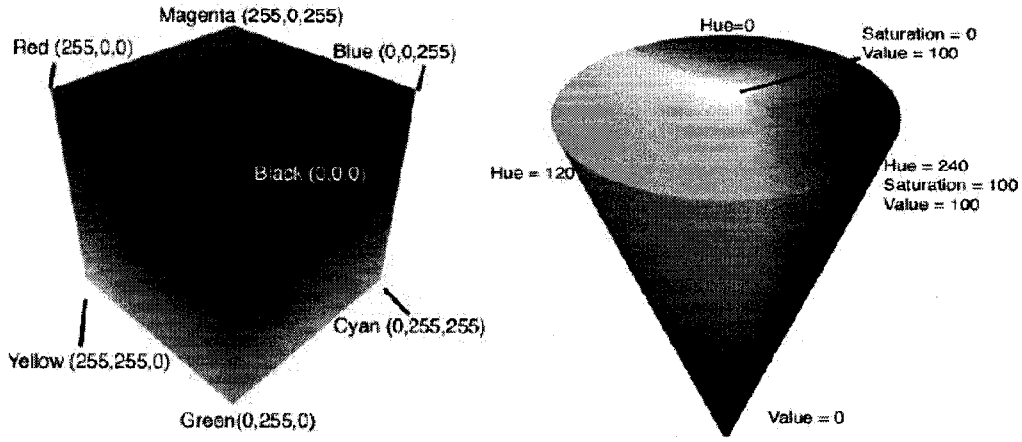

Figure 1.6 Black corner of RGB cube (left), HSV cone (right) (source: [21])

The three components of RGB color space are red, green, and blue. This system uses 256 codes for each R, G, and B channel; so, there are $256 \times 256 \times 256$ or 16 million distinct color codes. As it is shown at the right in figure 1.6, RGB is an additive color system; this means the colors are created by adding components to black: (0, 0, 0) [20]. In HSV (Hue, Saturation, Value) color system, as it is shown in figure 1.6 (left), colors are encoded by separating out intensity or value from hue and saturation [20]. This

system uses 360 codes for H. HSV color space is similar to the way humans perceive color. Moreover, hue parameter of this system is partly invariant to the changes in illumination. Therefore, many works used HSV color space for object detection, shadow detection, and tracking purposes. For example, in works [22, 23], authors proposed an algorithm to distinguish moving regions belonging to a shadow and/or an object, using the fact that the pixels of areas covered by shadows has significant change in lightness (V) without a great modification of the color information (H and S).

Color information can be described with the color histogram, color coherence vector, color correlogram, and color moments. Color histogram and color correlogram are the two best-known color descriptors for tracking systems.

-Color histogram

For the color histogram (figure 1.7), first the color space is divided into K intervals, then each color interval is assigned to a bin of the histogram. To build the histogram of a region in the image, the number of occurrence of region pixels belonging to each color intervals is counted and is assigned to the corresponding bin in the histogram. This structure is easy to build and it can contain global and local distribution of colors; moreover it is robust to scale changes, partial occlusion, rotation, and translation.

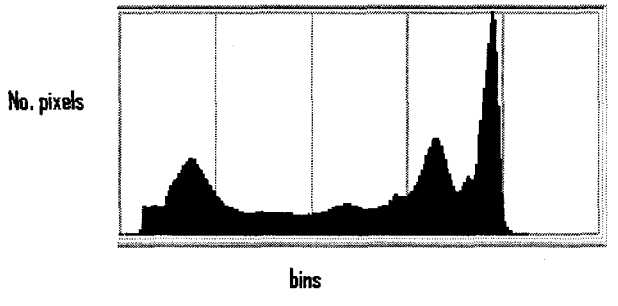

Figure 1.7 Color histogram

Clearly, the more bins a color histogram contains, the more discriminative power it has, since the color intervals are smaller. But on the other hand, computational complexity of comparing two color histograms will be increased. Adjusting the number

of k colors for histogram bins is color quantization. For example, in the work [24], the color space of object is partitioned using k-means clustering. Accordingly, histogram bins are determined. Some other studies have not used clustering for quantization; they just simply divided the color space to equal interval to find the number of histogram bins.

-Correlogram

Correlogram is another color descriptor. It contains not only the information of color distributions of object pixels, but also the spatial correlation of each pair of colors. This characterization helps to discriminate two objects better when they have similar color distribution but different spatial color distribution. Since correlogram contains spatial information of region's color, it can be categorized as texture features. A color correlogram is a table indexed by color pairs, where the k^{th} entry for (i, j) specifies the number of occurrences of color j at a distance k from a pixel of color i in the region belonging to the object [25]. Hence, a correlogram is similar to a histogram, but instead of counting the number of occurrences of a given color, it counts the number of occurrences of pairs of colors having a given spatial relationship expressed by distance k.

In the works of [15, 26], they used correlogram as one of the features to represent the object in a tracking system.

1.2.1.3 Texture

Texture is another region-based feature used for object representation. Texture description is an approach to quantify the object region properties such as smoothness, coarseness, and regularity [27]. In the remainder of this subsection, some of the most used texture descriptors in object tracking are described.

- Co-occurrence matrix

In a grayscale image, a co-occurrence matrix is a two-dimensional array, which saves the number of times that a gray intensity value i co-occurs with gray intensity value j inside a particular spatial distance k in the object region (It is a grayscale correlogram).

In figure 1.8, a grayscale image and its corresponding co-occurrence matrix is presented by considering the spatial distance $k = (1,3)$ for a 2D image. As an example, in the work presented by [17], an extended tracking system using region, boundary, and shape information is designed for both color and infrared (IR) video sequences.

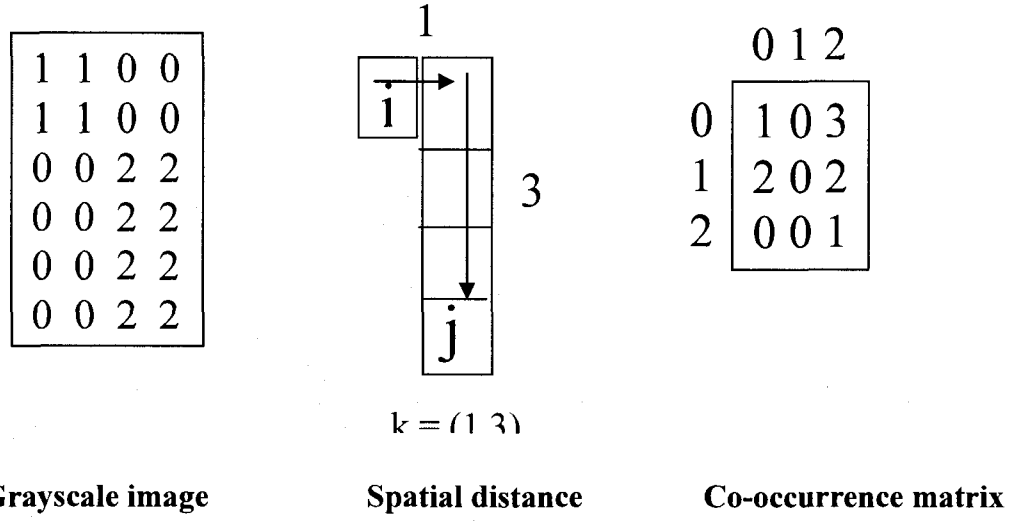

Figure 1.8 Co-occurrence matrix

- Local Binary Partition

The other approach to represent texture feature is Local Binary Partition (LBP).

In this approach, for each pixel p, a 8-bit number ($b_1b_2b_3b_4b_5b_6b_7b_8$) is generated using eight neighbors around p where $b_i = 0$ if neighbor i has value less than or equal to $p's$ value and otherwise $b_i = 1$ [20].

In figure 1.6 (a), a pixel with its 8 neighbors is illustrated (number assigned to each pixel is color intensity of that pixel), and in figure 1.6 (b), its corresponding binary number is generated. LBP is used in the work [28] as a texture feature for grayscale, because in grayscale color space, LBP is robust to illumination changes.

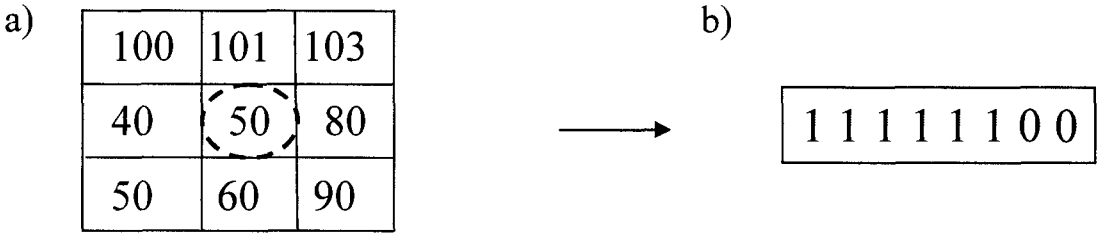

Figure 1.9 Local Binary Partitions (LBP)

- Gabor filter

The Gabor wavelet representation of an image is the convolution of the image with a bank of Gabor filters. This filter exhibits desirable characteristics of spatial locality, scale, and orientation selectivity. These characteristics are used as features for object representation. To obtain the Gabor representation of an image, all the filters in the filter bank need to be convoluted with the image which causes a great computational complexity. A filter bank consisting of Gabor filters with different orientations and scales is normally used for feature extraction. Gabor filters are local features invariant to rotation, scaling, and illumination changes [29]. Gabor filters come in pairs [30]. Its equations are defined as

$$G_{symetric}(x,y) = \cos(k_x x + k_y y)\exp-\left\{\frac{x^2+y^2}{2\sigma^2}\right\} \quad \text{and} \qquad (1.6)$$

$$G_{antisymetric}(x,y) = \sin(k_x x + k_y y)\exp-\left\{\frac{x^2+y^2}{2\sigma^2}\right\} . \qquad (1.7)$$

In equations 1.6 and 1.7, (k_x, k_y) give the spatial frequency to which the filter responds most strongly, and σ refers to scale. In [29], the features used for object representation are detected according to Harris corner detector and the feature descriptor is computed as the Gabor filter responses. This means at the locations specified by the

Harris corner detector, the feature vector is extracted by taking the response of the image with the Gabor filters in the filter bank. The filter bank that they used had 6 orientation variations and 4 scale variations which result to 24-dimensional feature vector. In [31], to represent the object; first the Gabor filter bank is applied to filter difference image between two consecutive frames. Then on the high magnitude moving area, a number of points are positioned and at last a k-mean classification is used to classify each point set as an object.

- *Edges*

An edge is a set of connected pixels that lie on the boundary between two regions [27]. Object edges contain useful information for object tracking. The main difference between boundary and edge is that edge is a local feature but boundary is a global feature of object. One of the common edge descriptor is edge orientation histogram. As an example look at the work *of Changjiang et al.* [32]. In this work, edges are extracted by Sobel operator, and then the edge orientation histogram is used as one of the descriptors to represent the object. To build an edge orientation histogram first the horizontal and vertical Sobel operators are applied on the image pixels to extract edges, and then the strength and the orientation of edges are calculated as

$$S(x,y)=\sqrt{\mathrm{G}_{\mathrm{x}}^{2}(\mathrm{x},\mathrm{y})+\mathrm{G}_{\mathrm{y}}^{2}(x,y)} \quad \text{and} \tag{1.8}$$

$$\theta(\mathrm{x},\mathrm{y})=\arctan(G_{y}(x,y)/G_{x}(x,y)) \ , \tag{1.9}$$

where G_x is the horizontal gradient and G_y is the vertical gradient. At last, the edges are counted into K bins based on their strengths and orientation. Figure 1.10 (right side) shows the polar plot of object edge orientation histogram.

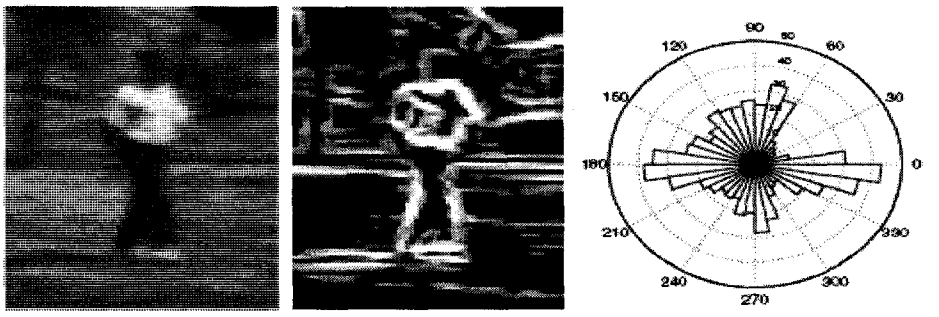

Figure 1.10 Original image (left), detected edges (middle), polar pot in Θ and R (right) (source:[32]).

Edge orientation feature is not invariant to rotation and scale; therefore, it is not a reliable representation for some image retrieval applications. But for tracking, the scale and direction of objects in two consecutive frames do not change dramatically, and hence this feature might be appropriate.

1.2.1.4 Motion

Motion is a feature, revealing the dynamics of scenes by relating spatial image features to temporal changes [33]. Optical flow is a motion descriptor which approximates the motion of objects in sequences of 2D images. It can be used as input for processing such as motion detection (object detection), velocity estimation for object tracking, and autonomous navigation. Optical flow is defined as vectors which represent the motion of object regions or pixels within frames. It is computed by matching pixels based on their gradients in space and in time. It can be computed under the assumption that the intensity of pixels near corresponding points is relatively constant [20]. In [34], the optical flow is used as features, under a non-priori training feature model framework, to detect and track moving objects. In this work [34], the algorithm tracks a set of feature points based on optical flow. At each frame the position of extracted feature points for next frame are predicted and, by receiving the new observations, are corrected. In figure 1.11, the light points are feature points and dark short lines are optical flow vectors.

Figure 1.11 Optical flow-based tracking results (source: [34]).

1.2.2 Model-based representation

In the model-based object representation for tracking systems, a priori model usually is built before starting the tracking. This means the objects are tracked by matching projected object models, produced with prior knowledge. This approach is mostly used in the systems designed for classification of different types of moving objects and high-level object behavior interpretations. For example, it is widely used for human and vehicle motion analysis [35-37]. For a tracking system designed for surveillance applications, having a precise model of object may lead to better data association results. But on the other hand, the computation complexity of data association will be increased. Moreover, a model-based object representation has a training step so it is not applicable for online surveillance systems. Since in most works, models are built for human and vehicles, a categorization of model-based object representation for human and vehicle can be as Skeletal, Active contour, and Volumetric models which are described in the following sections.

1.2.2.1 Skeletal models

Skeletal object model or stick figure is another way to represent objects and it is widely used for human modeling in motion analysis applications. This model represents the parts of a human body (e.g. head, torso, arms, and legs) as sticks and links the sticks with

joints. The stick figure model indicates the order of the motion and spatial relationships between the body parts [38]. For example, in [39], a system is designed to recognize continuous human activities in lateral view such as sitting down, bending, and standing up. For training, a dataset of single action video sequences is used. To compare models with image sequences, first segmentation and skeletonization (medial axis transform) is performed to extract stick figure. Skeletonization consists of finding the local symmetry axes of a shape. Skeletonization is used to obtain three main components of body: torso, upper component of leg, and lower component of leg. After this, three angles of inclination for torso, upper component of leg, and lower component of leg are defined as a feature vector for each frame. Then beginning and termination of the action is found by using the angle information and by comparing with training data set. At last, actions are classified using the fact that angles traverse the characteristic path during the execution of each action.

1.2.2.2 Active contour models

The active contour based object representation, describes an object by its outline as bounding contour and updates this contour dynamically in successive frames by minimizing an energy function usually based on the gradient on the outline of the object. We categorized this object representation as the model-based representation, because the initialization of tracking object with active contour usually is not automatic. The advantage of this method compared to the region-based object representation is that it reduces complexity of object matching, but on the other hand its sensitivity to occlusions and shadows is a problem. This is because the shadow connected to the object can change the appearance of object boundary [40].

1.2.2.3 Volumetric models

The volumetric model of an object is a spatial representation of the object in a 3 dimensional space. Compared to 2D models, 3D models of an object are less sensitive to camera's angle and lightening changes. But the main drawback of the 3D model is that it

requires more parameters and leads to more expensive computations during the matching process for tracking purposes [5]. In [41], for spatial modeling of various types of vehicles, 12 length parameters are used. In their work, matching between the model and the image data is performed first by projecting 3D model of cars (Models are built with lines but in three dimensions) in a 2D space. In this process some length parameters are hidden (because they are not visible in the view of camera); then the image edge segments are extracted; at last correspondence between model and image edge segments is established. In [42], an algorithm is designed to track an arm. A 3D model of an arm is built by 2 truncated right-circular cones for upper and lower part of arm and 2 spherical joints are used for shoulder and elbow joints. Authors in [42] imply one degree of freedom to the cone that belongs to the forearm and three degrees of freedom to the upper arm; therefore their model contains 7 parameters: 3 lengths for forearm, upper arm, and hand and 4 diameters belonging to 2 limbs at each end. In their work, an extended Kalman filter for recursively predicting the appearance of the arm in the image is used by using current estimate of the arm position and the difference between predicted and actual image are also used as an error measurement for Kalman filter estimator.

1.3 Data association (Tracking strategies)

The task of establishing correspondence between the targets (tracking object) across frames is known as data association or tracking strategy. The ultimate goal of the tracking strategy is to generate the trajectory of an object tracked over time by locating its position in every frame of a video sequence. Data association can be formulated as a combinatorial motion correspondence problem to find the trajectory of tracked objects. For instance, with n frames and m objects in each frame the number of possible trajectory sets is $m!^{(n-1)}$ which is a really large number even for a small number of frames and objects [43]. For this huge number of possible trajectories, trying all possible trajectory sets is impossible. Also in the cases dealing with short and long term disappearing objects, and occlusions finding a measurement to make decision about correct trajectories is problematic.

To solve the data association problem two global categories of approaches exists: 1) the approach based on sequential logic, which make decisions at each frame and when decisions are made, they are unchangeable. 2) The approach based on deferred logic, which considers a set of observations before each decision. This approach in the extreme case may lead to an offline and batch processing [5].

In the context of visual surveillance, the tracking strategies are used from either sequential or deferred logic and are categorized as shown in figure 1.12.

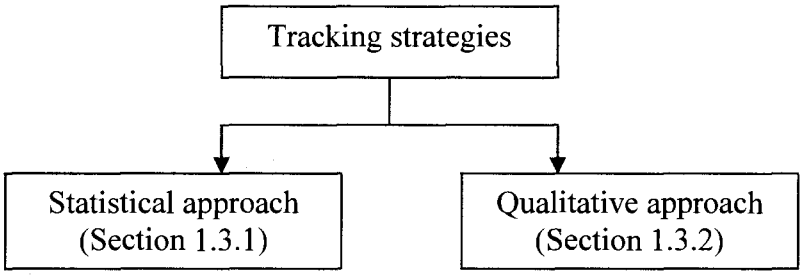

Figure 1.12 Tracking strategies

1.3.1 Statistical approaches

The statistical tracking approach resolves the data association problem by taking measurements and modeling uncertainties during object state estimation, which is appropriate for tracking objects in a clutter. Two main limitations of this approach are: 1) in some statistical models, the assumptions that the measurements are distributed normally around their predicted position may not hold (non-Gaussian distribution), 2) since statistical techniques model all events as probabilities, these techniques typically have a large number of parameters [44]. In the remainder of this section, the probabilistic tracking framework is described and three of the most used statistical methods for visual tracking are presented.

1.3.1.1 Probabilistic tracking framework

The probabilistic framework formulates tracking problem as a state space approach. This framework consists of two parameters: 1) state; which is the features used to represent object such as object's location or velocity (X^t; state of object in t). 2) Measurement; which is the observations obtained by object detection module such as object position (Z_t: measurement in t). To change the state, a transition equation is defined as

$$X^t = f^t(X^{t-1}) + W^t, \tag{1.10}$$

where f is the function of system (transition function between *t-1* and *t*) and W^t is the noise parameter (white random noise), and to relate measurement and state, a measurement equation (h) is defined as

$$Z_t = h^t(X^t, N^t), \tag{1.11}$$

where N^t is the noise parameter.

The objective of tracking is to estimate the state X^t given all the measurements $Z_{1:t}$; this means $P(X^t \mid Z_{1:t})$. The theoretically optimal solution to solve state estimation is provided by recursive Bayesian filter [11].

1.3.1.2 Kalman filter

Kalman filter is a state space recursive Bayesian method appropriate for solving time-series problems such as tracking. For the tracking problem, this filter recursively calculates some degree of belief in the state X^t, which is the state used to define the characteristic of the tracked target, at time t, given the data $Z_{1:t}$ (the measurements up to time t). This method is more appropriate for straight-through tracking approach. The Kalman filter consists of two main phases: predict and update. The prediction phase uses the system model to predict the state posterior probability density function (*pdf*) before

having next measurement, and the update phase uses the latest measurement to modify the prediction *pdf*. Prediction phase uses the state model to predict the new state of target with

$$\bar{X}^t = D\bar{X}^{t-1} + W \text{ and} \tag{1.12}$$

$$\bar{\mathrm{E}}^t = D\mathrm{E}^{t-1}D^T. \tag{1.13}$$

In equation 1.12 and 1.13, $\bar{X}^t$ and $\bar{\mathrm{E}}^t$ are state and covariance prediction at time t. D is a transition matrix which defines relations between state variables at time t and $t-1$. Q is the covariance of noise W. The correction phase uses current observations Z^t to update the object's state with

$$K^t = \bar{\mathrm{E}}^t M^T \left[M\bar{\mathrm{E}}^t M^T + R^T\right]^{-1} \text{ and} \tag{1.14}$$

$$X^t = \bar{X}^t + K^t\left[Z^t - M\bar{X}^t\right] \text{ and} \tag{1.15}$$

$$\mathrm{E}^t = \bar{\mathrm{E}}^t - K^t M\bar{\mathrm{E}}^t. \tag{1.16}$$

In equations 1.12, 1.13, and 1.14 M is the measurement matrix, K is the Kalman gain. The Kalman filter assumes posterior density is always Gaussian [11]. As an example of using Kalman filter for tracking purposes, in [45] a system for monitoring pedestrians and cyclists is designed. Their work has two steps: 1) tracking targets and 2) classifying targets into pedestrians and cyclists. In the tracking step, the object's centroid is used as a feature with Kalman filter to track the object. The horizontal and vertical coordinate of the centroid is decomposed as two independent motion vectors and they are modeled separately. In [45], it is assumed that the object moves with constant velocity with zero-mean white Gaussian acceleration error. The state of tracker is predicted using two models for horizontal and vertical components as

$$\begin{bmatrix} x_{i+1} \\ \dot{x}_{i+1} \end{bmatrix} = \begin{bmatrix} 1 & T \\ 0 & 1 \end{bmatrix}\begin{bmatrix} x_i \\ \dot{x}_i \end{bmatrix} + \begin{bmatrix} T^2/2 \\ T \end{bmatrix} v_{x,i} \text{ and} \tag{1.17}$$

$$\begin{bmatrix} y_{i+1} \\ \dot{y}_{i+1} \end{bmatrix} = \begin{bmatrix} 1 & T \\ 0 & 1 \end{bmatrix} \begin{bmatrix} y_i \\ \dot{y}_i \end{bmatrix} + \begin{bmatrix} T^2/2 \\ T \end{bmatrix} v_{y,i} \quad . \tag{1.18}$$

In the equation 1.17 and 1.18 T is the time step between frames and v_{xi} and v_{yi} are white Gaussian acceleration errors. The observation is selected based on a geometric distance from predicted object centroid and structural similarity; at last the best match is used as a new observation of object's centroid and it is used to update the state model of object.

1.3.1.3 Particle filter

The particle filter or sequential Monte Carlo method is another recursive statistical approach that recently is used in many tracking researches. The most important advantage of the particle filter compared to Kalman filter is that it is not limited by the assumption that the state variables have a Gaussian distribution. This method is more appropriate for straight-through tracking approach. In this method *pdf* of system at time t is approximated by a set of N weighted samples (particles) $\{x_t^i, w_t^i\}_{i=0}^N$ [46], each sample is an estimation about state of system which can be the location of object with its scales (height and width) . The main steps of this algorithm are as follows [47]:

1) Draw N samples from importance distribution $q(x_t \mid x_{1:t-1}, z_{1:t})$;
2) Evaluate the importance weights for N samples, z is observation at time t;

$$\hat{W}_t^i = W_{t-1}^i \frac{p(z_t \mid x_t^i) p(x_t^i \mid x_{t-1}^i)}{q(x_t \mid x_{1:t-1}, z_{1:t})} \tag{1.19}$$

3) Compute normalized importance weight;

$$W_t^i = \frac{\hat{W}_t^i}{\sum_{i=1}^{N} \hat{W}_t^i} \tag{1.20}$$

4) Compute an estimate of effective number of particles;

$$\hat{N}_{eff} = \frac{1}{\sum_{i=1}^{N} (W_t^i)^2} \tag{1.21}$$

5) If the effective number is less than a fix threshold then perform re-sampling; Draw P particles from the current particle set with probabilities proportional to their weights. Replace the current particle set with this new one.
6) Go back to step 1

The particle filter tracking algorithms often uses contour, color, or appearance models. As an example in [32], color feature and edge orientation histogram are used to model observations. The state of system is $X = (x, y, s_x, s_y)$, where x, y indicate the location of target and s_x, s_y are the scales (height and width) in horizontal and vertical directions. The state of system at time t is approximated by a set of samples. The samples are generated by using quasi-random sequence generator which draws samples from a normal distribution. The weight of each sample is observation likelihood related to each sample.

The most important advantage of particle filter is that, the tracking system does not need a separate object detection module like background subtraction. And, its limitation is that initialization of system is not simple and most of time is not automatic.

1.3.1.4 Multiple hypothesis tracking

Multiple hypothesis tracking (MHT) is an iterative statistical tracking algorithm, appropriate for tracking multiple objects with interactions and occlusions in a merge-split tracking approach. This approach is based on deferred logic for data association; this means correspondence decision is deferred until several frames have been examined. The final track of the object is the most likely set of correspondences over the time period of its observation. For occluded targets, this approach results in better data association

compared to the methods that use a sequential logic. The limitation of this approach is that it is computationally exponential both in time and memory. The first MHT as an algorithm for tracking multiple targets is proposed by Reid [10]. The basic approach of *Reid's* MHT is to generate sets of hypotheses for all possible data association and searching all possible paths between measurements and their origins. His approach is measurement-oriented; this means every possible target is listed for each measurement. Measurement-oriented approach is more appropriate for track initiation, because when no target is listed for a measurement, the measurement can be assumed to be a new track. His approach has four stages which are shown in figure 1.10.

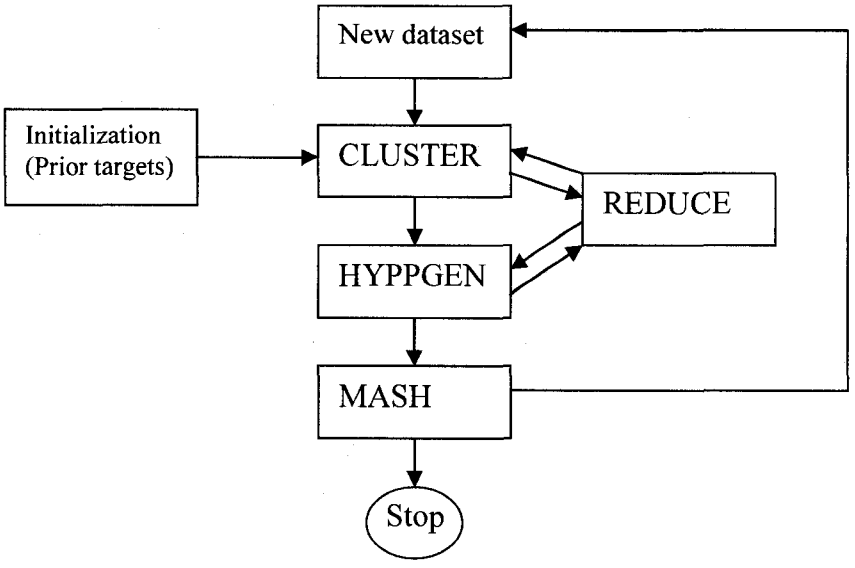

Figure 1.10 MHT stages

1. CLUSTER: When new measurements (the detected moving region in the scene) are available, this stage identifies which measurement is associated with which target (target state is estimated from each data association hypothesis using Kalman filter). A cluster is a set of measurements and targets that are associated to each other during period of time, in fact tracking problem may compound more

than one cluster that are solved independently which reduce computational requirement .

2. HYPGEN: In this stage, new sets of hypotheses (data association between current and previous data) for each cluster are formed and their probabilities are calculated. This means as each measurement is received; probabilities are calculated for the hypotheses from the measurement. Three types of hypothesis exist: 1) Hypothesis from previously known targets in a target file, 2) Hypothesis as a new target 3) Hypothesis as a false measurement.
3. MASH: In this stage, to minimize computational requirement, the hypothesis matrix of each cluster is simplified. The tentative targets with unit probabilities are transferred to confirmed targets and a new cluster is created for them. In this way, the entire set of targets and measurements are divided into clusters that are solved independently.
4. REDUCE: In this stage, to reduce the number of hypotheses, unlikely hypotheses are eliminated and hypotheses with similar target estimates are combined.

The limitation of *Reid's* MHT is its exponential growth of hypotheses when a new observation is received. To improve *Reid's* algorithm *Cox et al.* [48] proposed an efficient implementation of *Reid's* algorithm in which the *K*-best hypotheses are determined in polynomial time by removing unlikely hypotheses. Also the experimental result of this work [48] shows that real-time MHT solution to motion correspondence problem like tracking is possible.

In [49], a MHT is proposed to find the trajectories of objects. In this work, the graph structure is used to represent multiple object tracking. In the graph, nodes represent object detection results (detected objects in the scene). The state of each detected object is defined as an object detection probability, object size, location, and appearance. Each link (connection probability) in the graph is computed based on the position closeness, size similarity, and appearance similarity between two nodes. In this work each

hypothesis consists of a fixed number of objects and their trajectories (the same as *Reid's* MHT [10]). Then, the likelihood or probability of each generated hypothesis is computed according to the connection probability, the object detection probability, trajectory smoothness, and image likelihood computation. At last, hypotheses are ranked based on their likelihood values and to avoid computational explosion in the graph extension, only a limited number of likely hypotheses are kept and the graph is pruned accordingly.

1.3.2 Qualitative approches

All the tracking approaches which are not probabilistic are categorized as qualitative approach. These approaches are usually simpler and have fewer parameters compared to the statistical approaches. Qualitative approaches define a cost function to associate objects in two consecutive or more frames using a set of motion constraints. The main advantage of this approach is their flexibility, because they permit an easy integration of a priori information for constraining a complex problem. It is necessary to mention that there are some methods such as mean-shift that use a prediction step to predict the object in the next frame but their method of tracking is qualitative. Mean-shift is one of the known qualitative tracking techniques which also uses object segmentation and is presented in the next subsection.

1.3.2.1 Mean-shift

Mean-shift tracking method, which is also known as a kernel-based tracking method, is composed of three steps performed at each two consecutive frames: 1) Start from the position of the model in the current frame 2) Search in the model's neighborhood in next frame 3) Find best candidate by maximizing a similarity function. This method is appropriate for straight-through tracking approach. This process is repeated for each pair of frames. In [50], a mean-shift tracking algorithm is proposed. In this work [50], the object model is defined as the weighted histogram computed from a circular region and the similarity function is defined by Bhattacharya coefficient as

$$\sum_{i=1}^{b} P(i)Q(i) \quad , \tag{1.22}$$

In equation 1.22, b is the number of histogram bins, P is the target histogram, and Q is the candidate histogram. To find the best candidate in each iteration, mean-shift vector is computed such that the histogram similarity is increased. This process is repeated until convergence is achieved.

CHAPTER 2 METHODOLOGY

Tracking is a combinatorial motion correspondence problem with ultimate goal of finding trajectories of the targets in FOV of camera(s). An efficient structure for representing a tracking problem and analyzing spatial and temporal information of targets is a graph. Given a video with n frames and m objects in each frame, representing targets using a graph-based approach requires finding m best paths among $m!^{(n-1)}$ different possible paths (possible trajectory sets) in the graph (Figure 2.1).

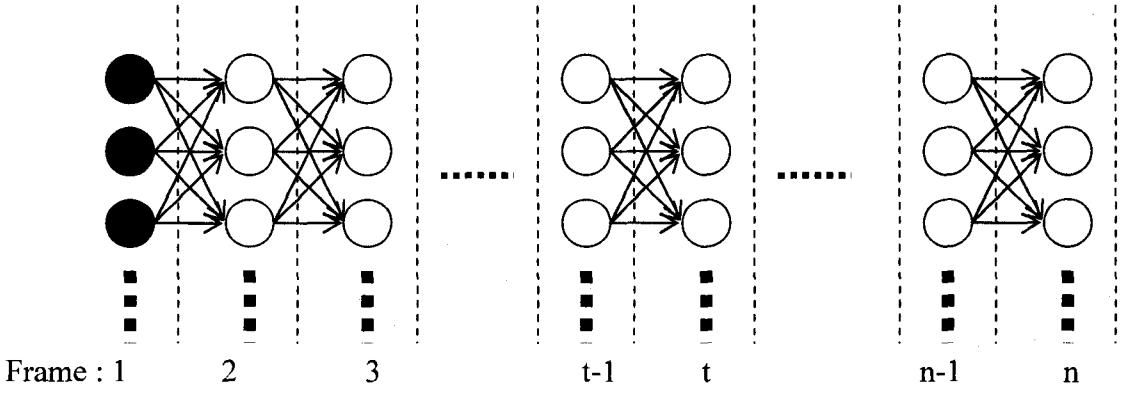

Figure 2.1 Graph representation of tracking

In figure 2.1, a graph representation of tracking is shown. The number of possible paths increases exponentially as the number of frames is increased. This suggests that this approach in its naïve form is not an efficient framework for tracking. In this chapter, we present an online multiple object tracking algorithm based on multiple hypothesis approach to handle uncertainties caused by the inter-object occlusion(s). We use a graph-based representation. This algorithm has two main features: 1) summarizing the graph representation for tracking 2) handling uncertainties in data association.

The first feature of the algorithm is a reduction of the number of edges in the tracking graph by the matching process which results in a simpler graph named event graph. This is done to reduce the search space of data association only for the situations in which uncertainties in data association exist. For example, splitting is one of these situations as it was discussed earlier in the previous chapter.

The second feature of our algorithm could be explained by the fact that the global information of several frames is used for data association in the event graph rather than information of two consecutive frames. The advantage of this modification is that the average visual features of an object during data association of several frames in occlusions is more reliable and stable compared to the visual features of object in one frame. The data association is done by building a hypothesis graph to generate hypotheses for associating targets and to choose the best set of association as the final trajectory result. To adapt our algorithm for online applications, the best associated label of each target is found at each frame, and the correction of wrongfully associated labels is done by receiving more information in later frames.

Pre-processing of this tracking system is performed by background subtraction to detect the blob at each frame, shadow elimination to delete the shadow regions from moving blob, a size filtering to delete very small blobs which are usually noises and are detected by background subtraction, and finally color conversion to convert the image pixels from RGB to HSV. The HSV color space is less sensitive to lighting variations and well suited for our experiment.

The remainder of this chapter is organized as follows; in section 2.1 the features used in our algorithm for representing an object in the image are presented and in section 2.2 data association strategy of our tracking system is described.

2.1 Model of object

In each frame the existing blobs in the FOV of camera are represented using information of the blob's centroid, size, bounding box (i.e. a rectangle around the blob region), color histogram, velocity, and state (i.e. one of the possible events that object can be in, such as merging and entering). The state of an object and its velocity are determined after the matching process which is explained later in this chapter. The previously mentioned visual features are mainly used because of their discriminatory abilities. These features are described in the following paragraphs.

In figure 2.2, the left image is a moving object and the right image is the extracted blob after performing background subtraction.

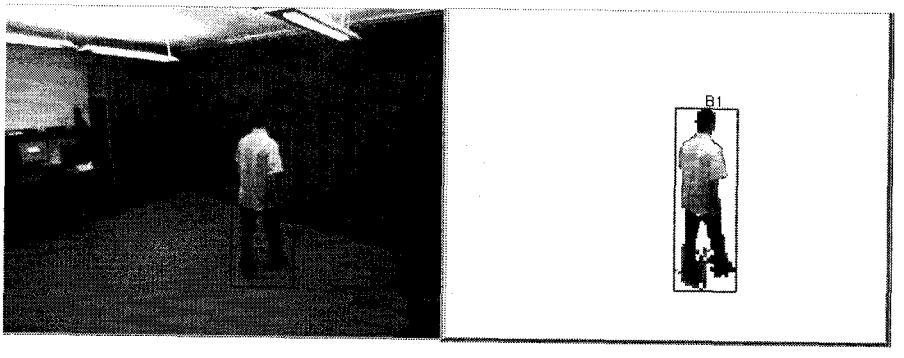

Figure 2.2 An extracted blob and its bounding box.

The feature vector used to represent a blob "B" is defined as

$$B = (s, c, b_1, b_2, b_3, b_4, h, d_x, d_y, st), \tag{2.1}$$

where s is size of the blob or number of pixels in blob regions which is formulated as

$$s = \sum_{i=1}^{N} \sum_{j=1}^{N} X_i^0 Y_j^0 , \tag{2.2}$$

In Eq. 2.2, N is the number of blob region pixels, X_i, Y_j are coordinates of each pixel in blob region, and c_x, c_y are the centroid coordinates of the blob's region pixels and are defined as

$$c \rightarrow c_x = \frac{\sum_{i=1}^{N} X_i}{s}, c_y = \frac{\sum_{j=1}^{N} Y_j}{s}, \tag{2.3}$$

where c is centroid of blob with coordinate $(c_x, c_y), N$ is the number of blob region pixels, and X_i, Y_j are coordinates of each pixel in blob region.

In the feature vector, b_1, b_2, b_3, b_4 are the coordinates of the four corners of the blob's bounding box and h is the normalized color histogram of the object in HSV color space. This histogram is quantized in 54 equal size bins and is represented as

$$h(i) = \left|\left\{\forall x, y \middle| (x, y) \in \text{Blob region} \cap I(x, y) \in ith \text{ color bin}\right\}\right|, \; i = 1,...,54, \tag{2.4}$$

where $h(i)$ is the ith bin in color histogram and $I(x, y)$ is the color intensity of pixel (x, y). Furthermore, d_x, d_y are the moving distance in X and Y of the blob's centroid in two consecutive frames. It is calculated as

$$moving\ distance \rightarrow d_x = c_x^{t-1} - c_x^t \;,\; d_y = c_y^{t-1} - c_y^t \;, \tag{2.5}$$

where c_x^{t-1}, c_x^t are horizontal coordinates of the blob's centroid in current frame t and previous frame $t-1$ and c_y^{t-1}, c_y^t are vertical coordinates of the blob's centroid in frame t and $t-1$.

Finally, st is the event state of the blob in the current frame such as merging blob(s) and/or entering blob(s).

2.2 Data association

In our algorithm, data association is performed at each frame by repeating the matching process and updating and processing the event graph and the hypothesis graph. We have used a merge-split approach to handle interaction between the objects. In our tracking system, when a blob appears in the FOV of the camera, a track node is added in the event graph and a hypothesis node is added to the hypothesis graph. If during the tracking, a blob is merged (occluded) with other blobs, a track node is added in the event graph for the merged blob and it is connected with edges to all the occluding/occluded blobs nodes in the event graph. If a splitting happens, for all separated blobs from the group, nodes are added in the event graph and the hypothesis graph. In the event graph, edges are connected from the group blob's track node to all splitting blobs track nodes. In hypothesis graph, new initiated hypothesis nodes are connected to all the previous nodes (related parents) in the hypothesis graph with weighted edges (the weight is the likelihood of each pair of the new connected nodes). Finally, by processing the two graphs, the best label of the object in each frame and the best trajectory of each object while it is disappearing from FOV of camera are found.

In Figure 2.3, the steps of our algorithm for each frame are shown; these steps are repeated in each frame until in one frame no blob is detected in the scene.

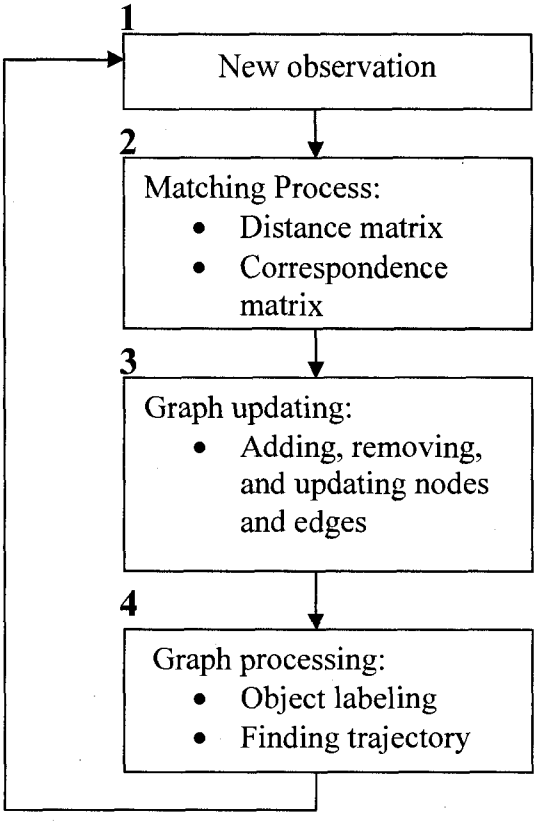

Figure 2.3 Data association steps

The remainder of this section is organized as follows. In section 2.2.1 the event graph is defined and in section 2.2.2 the hypothesis graph is defined also. In section 2.2.3 matching process is described. In section 2.2.4 updating hypothesis and event graphs are explained, in section 2.2.5 object labeling is presented, and finally, in section 2.2.6 finding trajectory of objects is presented.

2.2.1 Definition of event graph

The event graph consists of all identified blobs with their merging/splitting events. Its nodes are named track node and store appearance information of a blob (when blob is tracked) and its edges represent merging and splitting events among the blobs.

The attributes of each track node are: 1) a pointer to its corresponding hypothesis node (if it has one; because group blobs do not have hypothesis node), 2) feature vectors of the blob (Equation 2.1) during its tracking (one feature vector for each frame), and 3) its adaptive color histogram which is updated by receiving the blob's color histogram at each frame. The adaptive color histogram of a blob is the summation of its color histograms during tracking. This gives global color information of the blob. The adaptive color histogram is further explained later on.

Figure 2.4 shows an event graph. Here, the track nodes and edges represent the following events: blob B1 and B2 are merged into blob G1, which in turn, merge with blob B3 to form the merged blob G2. Then G2 is split in two blobs. This graph is built and updated using the events detected in the matching process such as entering (inserting track nodes for B1, B2 and B3), merging (inserting track nodes G1 and G2), splitting (inserting track nodes B4 and B5), corresponding, and leaving. The trajectory of each tracking target is determined by finding the best tracking path in the event graph. To find the best path for each tracking target, we utilize a hypothesis graph which is defined in the next subsection.

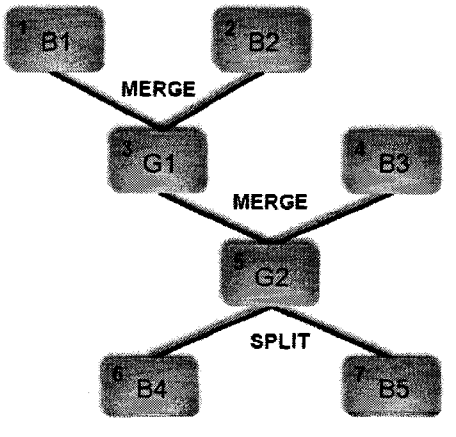

Figure 2.4 Example of an event graph. The number in the upper left corner of each node is a pointer to a hypothesis node.

2.2.2 Definition of hypothesis graph

Hypothesis graph is a directed weighted graph in the form of

$$G = (V, E, \omega). \tag{2.6}$$

The vertices of the graph (hypotheses nodes) simply corresponds to track nodes of the event graph belonging to entering blobs (blobs which appeared in scene) and split blobs (blobs which came apart from a group blob or a single blob). In the other words identified group blob (merged blob) do not have hypotheses nodes. This is because hypothesis nodes are used to solve the correspondence problem (data association) before and after a merging event. They are defined as

$$V = \{n_i\}_{i=1}^{m}, \tag{2.7}$$

where m is the number of hypothesis nodes. The edges are defined as

$$E = \{e_k\}_{k=1}^{n}, \tag{2.8}$$

where $e_k = n_i n_j$ is the edge between two hypothesis nodes n_i and n_j, and n is the number of edges. The weight of each edge which represents a hypothesis is defined as

$$\omega(n_i, n_j) = \left| AH(n_i) - AH(n_j) \right|, \tag{2.9}$$

where $AH(n_i)$ and $AH(n_j)$ are adaptive color histograms belonging to n_i and n_j nodes in hypothesis graph and $\omega(n_i, n_j)$ is the Euclidean distance between these two adaptive color histograms. Each node stores three different types of edges as three sets called *Parents*, *Children* and *Best hypotheses (BH)* defined as

$$Parents(n_i) = \left\{ \forall n_j \in V \middle| \exists e = n_j n_i \right\}, \tag{2.10}$$

$$Children(n_i) = \left\{ \forall n_k \in V \middle| \exists e = n_i n_k \right\} \text{ and} \tag{2.11}$$

$$BH(n_i) = \left\{ \forall n_j \in V \middle| Children_1(n_j) = n_i \right\}. \tag{2.12}$$

In Eq 2.10, *Parents* set of n_i are a subset of nodes V which have common edges (are linked) with n_i, where n_i is the end vertex of their common edges (links). That is, the parents are the source nodes. In Eq 2.11, *Children* set of n_i are a subset of nodes V which have common edges (are linked) with n_i, where n_i is the source vertex of their common edges (links). That is, the children are sink nodes. *Parents* and *Children* sets are ordered increasingly based on the weights of their edges. In Eq. 2.12, *BH* (Best Hypotheses) set of hypothesis node n_i is some parents nodes (n_j) of n_i, for which n_i is the first element in n_j's *Children* set; $Children_1(n_j)$ is the first element (element with index 1) in *children* set of n_j. *Parents* sets, *Children* sets, and *BH* sets of hypothesis nodes are utilized during object labeling and for finding trajectory which is explained in the next subsections.

In figure 2.5, an event graph with its related hypothesis graph is shown. In the left top corner of each node in the event graph there is an identifier to find its related hypothesis node in the hypothesis graph, for example for blob with identifier node 4 which is called n_4, the corresponding hypothesis node is H3 with the same identifier of 4. We note that blobs G1 and G2 do not have hypothesis nodes in the hypothesis graph because they correspond to merging events. In the hypothesis graph, each edge $W_{i,j}$ represents the weight of the edge between nodes H_i and H_j.

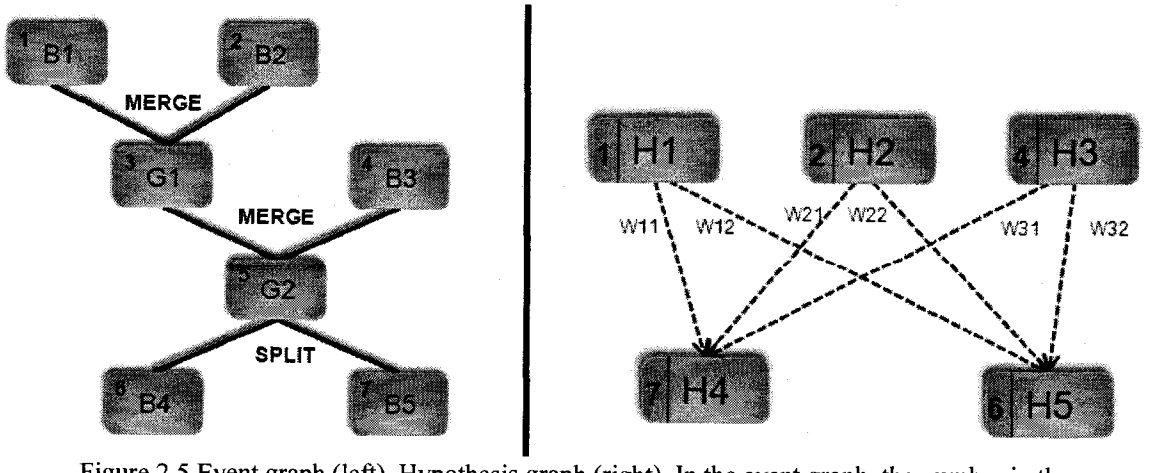

Figure 2.5 Event graph (left), Hypothesis graph (right). In the event graph, the number in the upper left corner of each node is a pointer to a hypothesis node (number at the left).

2.2.3 Matching process

In this step (step 2, figure 2.3) of our algorithm, we define a distance matrix and a correspondence matrix to detect events such as corresponding, merging, splitting, entering, and leaving. These events are used for updating graphs in the next step. Matching process is finding all the possible correspondences between blobs of two consecutive frames. Let us define the blobs in frame $t-1$ as

$$B(t-1)=\{B_1(t-1),B_2(t-1),...,B_i(t-1),...,B_N(t-1)\}, \tag{2.13}$$

where $B_i(t-1)$ is the feature vector of the ith blob in frame $t-1$ and N is the number of blobs. Similarly blobs in frame t are defined as

$$B(t)=\{B_1(t),B_2(t),...,B_i(t),...,B_M(t)\}, \tag{2.14}$$

where $B_i(t)$ is the feature vector of the ith blob in frame t and M is the number of blobs.

To represent all the possible corresponding blobs with their appearance dissimilarities, a distance matrix with the size of $N\times M$ (N is the number of blobs in frame $t-1$ and M is the number of blobs in frame t) is defined as

$$D_{B(t-1)}^{B(t)}(i,j)=\begin{cases} Distance\,(h_{B_i(t-1)}-h_{B_j(t)}) & overlapped\ bounding\ boxes \\ -1 & otherwise \end{cases}, \tag{2.15}$$

where $D_{B(t-1)}^{B(t)}(i,j)$ is the intersection color histogram distance between the ith blob in frame $t-1$ and the jth blob in frame t, if the two blobs bounding boxes in frames $t-1$ and t are overlapped. Otherwise, these two blobs cannot match each other and their corresponding element in the distance matrix is -1. This assumption is based on the fact that a blob should not move a long distance in two successive frames because of the frame rate of the camera, and hence its bounding boxes should overlapped. So the search space for matching process is reduced only to the blobs with overlapping bounding boxes. We chose the intersection color histogram distance because it is fast to compute

and robust to partial occlusion caused by inter-object occlusion or occlusion due to the scene structure. The color histogram intersection is defined as

$$Distance\,(h_{B_i(t-1)} - h_{B_j(t)}) = \frac{\sum_{k=1}^{K} min\,(h_{B_i(t-1)}(k),\ h_{B_j(t)}(k))}{\sum_{k=1}^{K} h_{B_i(t-1)}(k)} \quad . \tag{2.16}$$

In Eq. 2.16, the number of histogram bins is 54 (the number of bins determined experimentally) and the intersection distance is defined as one minus the histogram intersection which is normalized by the size of the blob in frame $t-1$. This means a small distance is representing similar color histograms.

Figure 2.6 shows an example in which 2 blobs are detected in frame *t-1* and 3 blobs in frame *t*. It also shows all possible correspondence between pairs of blobs in two consecutive frames.

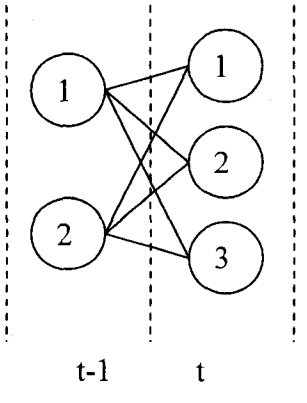

Figure 2.6 All possible correspondence between blobs in frames $t-1$ and t

For the example illustrated in figure 2.6, if we assume all the bounding boxes of blobs in the two consecutive frames are overlapped, the distance matrix (Equation 2.15) is defined as

$$\begin{bmatrix} D(1,1) & D(1,2) & D(1,3) \\ D(2,1) & D(2,2) & D(2,3) \end{bmatrix}_{2\times 3}. \tag{2.17}$$

Let us suppose that in figure 2.6, 2 in frame *t-1* matches 3 in frame *t,* and 1 in frame *t-1* matches 1 and 2 in frame *t*, then the distance matrix would be for example

$$\begin{bmatrix} 0.05 & 0.1 & 0.9 \\ 0.8 & 0.5 & 0.03 \end{bmatrix}_{2\times 3}. \tag{2.18}$$

In the next stage of matching process, by using the distance matrix and size information of blobs, a correspondence matrix [51] is defined to determine events and consequently update the event graph and hypothesis graph. The correspondence matrix has $N \times M$ elements (N is the number of blobs in frame $t-1$ and M is the number of blobs in frame t), the same as the distance matrix. First, all its elements are initiated to zero. The possible values in this matrix are 0, 1, and 2. To build this matrix, we define three possible situations:

- *Situation A*: if in the distance matrix, the minimum color intersection distance for i^{th} object in frame $t-1$ is j^{th} object in frame t AND the minimum color intersection for j^{th} object in frame t is i^{th} object in frame $t-1$ (Mutual correspondence).
- *Situation B*: if in the distance matrix, the minimum color intersection distance for i^{th} object in frame $t-1$ is j^{th} object in frame t XOR the minimum color intersection for j^{th} object in frame t is i^{th} object in frame $t-1$.
- *Situation C*: any other situation except A and B.

Now with above definitions, the correspondence matrix is built by

$$C(i,j) = \begin{cases} 2 \rightarrow situation\ A\ for\ (i,j) \\ 1 \rightarrow situation\ B\ for\ (i,j) \\ 0 \rightarrow situation\ C\ for\ (i,j) \end{cases} \tag{2.19}$$

The case where there is more than one non-zero element in a row i or column j means that there is an inter-object occlusion. We use size information of the blobs to finalize the values in the correspondence matrix. If a merging or splitting events happen, there is a dramatic change in the size of the blob. We assumed that object with no dramatic change in size cannot be a merging or splitting objects. So we change the corresponding elements in the correspondence matrix to zero. The possible events for blobs are entering/re-entering, leaving, merging, splitting, and correspondence. Using the correspondence matrix the events are detected as follows:

- *Entering*: Entering is detected for j^{th} blob in current frame t, if no blob in the previous frame $t-1$ corresponds to it. In other words, all the elements of *jth* column in the correspondence matrix are zero.
- *Leaving*: Leaving is detected for i^{th} blob in previous frame $t-1$, if no blob in current frame t corresponds to it. In other words, all the elements of i^{th} row in the correspondence matrix are zero.
- *Merging*: Merging is detected for j^{th} blob in current frame t, if more than one blob in previous frame $t-1$ corresponds to it. In other words, more than one elements of j^{th} column in the correspondence matrix are non-zero.
- *Splitting*: Splitting is detected for i^{th} blob in previous frame $t-1$, if more than one blob in previous frame $t-1$ corresponds to it. In other words, more than one elements of i^{th} row in the correspondence matrix are non-zero.
- *Correspondence*: Mutual Correspondence is detected between i^{th} blob in frame *t-1* and j^{th} blob in frame t if in i^{th} row and j^{th} column of correspondence matrix only element (i, j) is non-zero and is equal to 2.

For example illustrated in figure 2.6 with its distance matrix in Eq. 2.18 the correspondence matrix is defined as

$$C = \begin{bmatrix} 2 & 1 & 0 \\ 0 & 0 & 2 \end{bmatrix}_{2\times 3}, \tag{2.20}$$

where in C a splitting and a correspondence is detected as the events between frame *t-1* and *t*.

2.2.4 Updating graphs

Updating graphs (step 3, figure 2.3), depend on detected events in the matching process. Event graph and hypothesis graph are updated in each frame.

2.2.4.1 Entering

When the matching process determines that a blob in current frame *t* is an entering object, a new track node in event graph and a hypothesis node in hypothesis graph are added for that entering object.

2.2.4.2 Correspondence

When the matching process determines correspondence between two blobs in frame *t-1* and *t*, the track node in event graph belonging to the corresponding object is updated by adding its new appearance information for current frame *t* and updating its adaptive color histogram using

$$AH_{B(t)} = \sum_{i=1}^{K} \alpha AH_{B(t-1)}(i) + (1-\alpha)h_{B(t)}(i). \tag{2.2}$$

In Eq. 2.21, $AH_{B(t-1)}$ is the adaptive color histogram of blob B at frame *t-1*, K is the number of color histogram bins, $h_{B(t)}$ is the color histogram of blob B at frame t, and $0 < \alpha < 1$ is an adaptation parameter.

In fact, updating track node is equivalent to sequential data association for non-occluding blobs. This is based on the fact that if two blob regions in two consecutive frames are found to be similar, it is more likely that they are associated with the same object than if the two blob regions are separated by several frames. So in the normal situation when it is determined there is no possible occlusion. Sequential data association results to better data association rather than deferred logic data association.

2.2.4.3 Merging

When matching process determines that some blobs in frame *t-1* are merged in a single new blob in the current frame *t*, tracking of merging blobs in frame *t-1* is stopped and a new track node in event graph for group blob in frame t is created. It is characterized by a new feature vector and it is started to be tracked as any of the active blobs in the system.

2.2.4.4 Splitting

The most crucial event in a merge-split based data association is splitting. This is because uncertainties about identities of splitting blobs should be handled. This question should be answered at this point: Is splitting caused by separation of independent moving object regions or caused by object fragmentation? And if the answer to the previous question is splitting caused by independent objects, another question needs to be asked: Are the splitting objects already identified tracking targets or new tracking targets?

Figure 2.7 shows the occlusion handling steps. In step1, a possible splitting is detected, in step 2 track nodes are added to event graph, in step 3 fragmentation is distinguished from splitting, in step 4, if split blobs are remerged, splitting is ignored, in

step 5 if splitting is determined and splitting blob in frame *t-1* was not an identified merged blob, its hypothesis node is deleted because group node are not included in the hypothesis graph as they do not help for data association. In step 6, for split blobs in frame *t,* hypothesis nodes are added in hypothesis graph, and finally in step 7 hypotheses are generated.

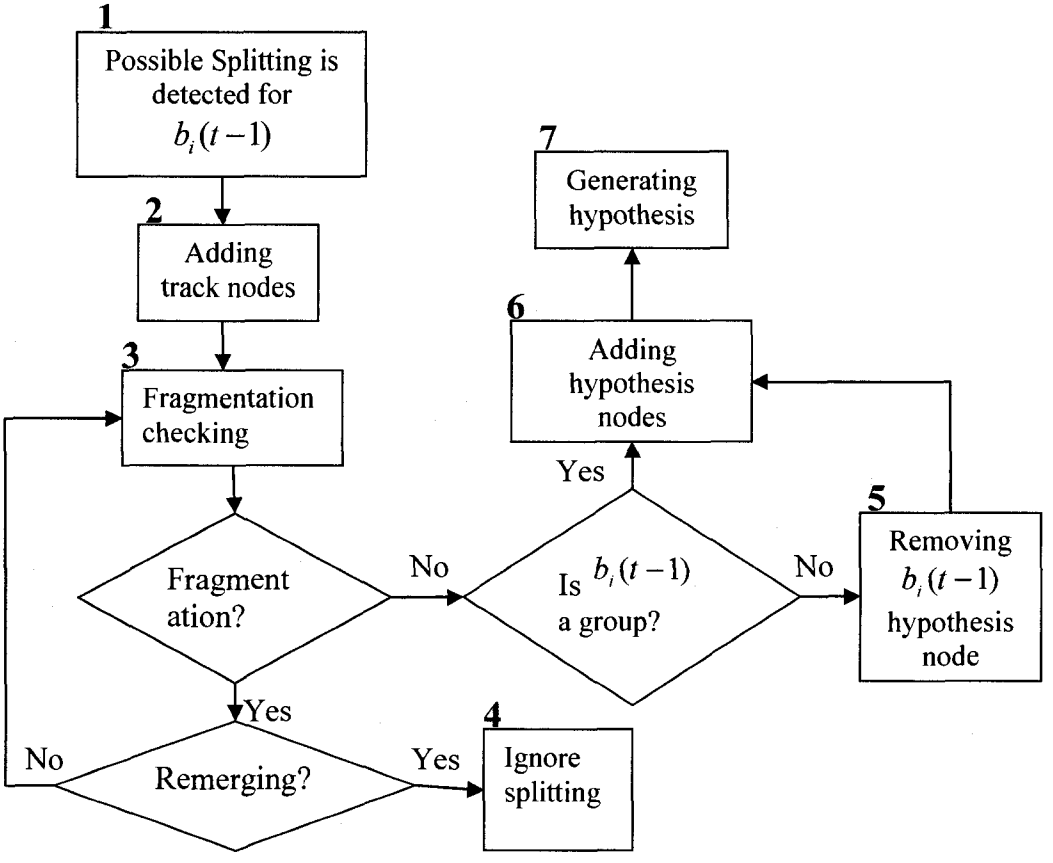

Figure 2.7 Occlusion handling steps

At each frame *t*, if matching process determined a splitting for a blob, for split blobs, track nodes are added in the event graph. Below the steps for occlusion handling are described.

a. Fragmentation checking

Because of the nature of background subtraction methods, whenever a moving object in the scene has a similar color as the background, some parts of the moving object may not be detected as the foreground. This results in detecting smaller blobs which are fragments of a moving object.

The goal of fragmentation checking (step 3, figure 2.7) is distinguishing between fragmentation and splitting events. So in this way, the tracking system does not track fragments as an independent objects. To do this, when the matching process detects a splitting event, the tracking system initiates track nodes for possible fragmented blobs. For four frames (determined experimentally), the system tracks the original blob as a whole, using summation feature vectors of all possible fragmented blobs. It also tracks all fragmented blobs individually. If within four consecutive frames after splitting all possible fragments are merged to one blob, it will be assumed that the event was fragmentation. In this case, splitting will be ignored (step 4, figure 2.) and the tracking of the original object will be continued. Otherwise after four frames, the fragmentation checking step distinguishes between fragmentations and splitting based on the fact that fragmented blobs belong to one target, are close in the image, and exhibit coherent motion over a few frames. In other words, the maximum differences in the average velocity between any pair of possible fragmented blobs from the same object must be less than a fixed threshold. This also suggests that the distance moved by the centroids of these blobs is relative to each other and much smaller than the overall distance moved by the centroid of the original object. If a fragmentation is detected, we continue tracking the original object and its fragments blobs. The fragmentation checking step (step 3, in figure 2.7) will be repeated four frames later. But if a splitting is detected, we stop tracking the original object and we will continue tracking the split blobs as individual objects.

b. Removing hypothesis node

Our algorithm has hypothesis nodes only for blobs appearing in the scene and split blobs, but it does not have hypothesis nodes for group blob (blobs which is known that they include more than one tracking target). This was justified previously. So, this suggests

that if a blob associated to a hypothesis node and splits in later frames, the blob will become known as a group blob. So its hypothesis node with its related edges will be deleted (step 5, figure 2.7). In this way, we keep the hypothesis node only for the individual object to have smaller search space in hypothesis graph and more accurate data association in our split-merge framework.

c. Generating hypothesis

For split blobs which passed fragmentation checking, hypothesis nodes are initiated (step 6, figure 2.7). Then for new initiated hypotheses nodes, hypotheses are generated by connecting directed weighted edges from all their *Parents* nodes in the graph to new initiated hypothesis nodes. *Parents* are all previous hypothesis nodes which possibly include a part of the trajectory of the same object that the split blob is also part of. Weight of each directed edge is the likelihood that source node and sink node of the edge are in the trajectory of the same blob and it is calculated as defined in section 2.2.2.

Figure 2.8 shows hypothesis graph belonging to two objects that occluded each other two times. As shown in figure 2.8, the group blob resulting from merging blobs belonging to nodes 1 and 2, and also 3 and 4, do not have hypothesis nodes in the hypothesis graph.

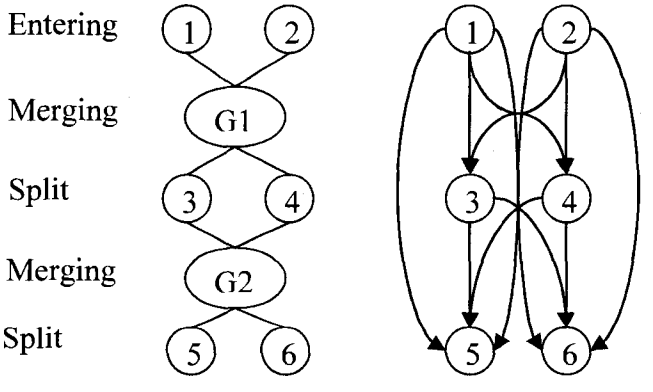

Figure 2.8 Generating hypotheses, event graph (left) and hypothesis graph (right).

In figure 2.8, hypothesis nodes 5 and 6 are two splits blobs for which their direct parents are nodes 3 and 4; these two split blobs have weighted edges not only from hypothesis nodes 3 and 4 (direct parents) but also from hypothesis nodes 1 and 2 (other previous nodes possibly in the trajectory).

After calculating each weighted edge, the *Children* set of its source node and the *Parents* set and *BH* set of its sink node are updated. After updating *Children* set of the source node, if the first element of this set is changed, consecutively some other nodes BH set should be updated (see definition of *BH* in section 2.2.2); based on the fact that the intersection of two *BH* sets for two different nodes should be empty.

The adaptive color histograms is used for generating hypothesis (likelihood between two nodes), because it gives the global color information of the blob during several frames and it helps to reduce the effect of dramatic changes in color histogram of blobs caused by short-time variation of lightening or shadows in some specific frames.

To better illustrate updating graphs, in figure 2.9, the event graph and hypothesis graph for the example presented in figure 2.6 with distance matrix in Eq. 2.18 and correspondence matrix in Eq. 2.20, is shown. Recall, that one of the to tracked blob splits. In figure 2.9, node identified by 1 and 2, correspond to blobs 1 and 2 at time *t-1* in figure 2.6. Node 3 and 4 in figure 2.9 correspond to blobs 1 and 2 at time *t* in figure 2.6. Finally, because there is a correspondence between blob 2 at time *t-1* and blob 3 at time *t* in figure 2.6, there is no new node added in figure 2.9.

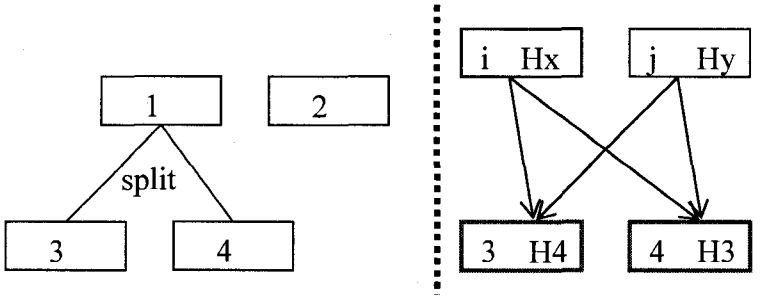

Figure 2.9 Example for updating graph. For the hypothesis graph, the numbers at the left correspond to the node numbers in the event graph.

In figure 2.9, for node 1 in event graph, a splitting event is detected and for node 2, a correspondence is detected. For updating graphs, for node 2 in event graph, only its visual appearance is updated. But for splitting node 1, track nodes 3 and 4 are added in event graph and also hypotheses nodes H3 and H4 are added to hypothesis graph and from their *Parents,* hypothesis are generated (Hx and Hy are shown as the hypothesis nodes (*Parents*) in frames before *t-1* only to show hypothesis generation). It's important to mention that figure 2.9 only shows a part of complete tracking graphs related to updates for events detected between frames *t-1* and *t.*

2.2.4.5 Leaving

When the matching process determined that a blob in frame *t-1* has left (disappeared from FOV of camera), its track node in event graph is deactivated. This results in its trajectory calculation by processing hypothesis and event graph which is described in section 2.2.6.

2.2.5 Object labeling

The goal of this step of our algorithm (step 4, figure 2.3) is to find the label to use as an identifier for each blob at each frame. For corresponding blobs in frames t-1 and t, the label of blob in frame t (current frame) is the same as the label of blob in frame t-1 (previous frame). For the merging, the label of merged blob (in frame t) is the labels of all merging blobs (in frame t-1).But for splitting, finding the label of split blobs in frame t (current frame) is done by processing in the hypothesis graph. To do this, we traverse the hypothesis graph bottom-up (from the latest frame) starting from split blob's hypothesis node n_i. A path set is initialized with

$$path_0(n_i) = \phi, \tag{2.22}$$

And it will be updated with

$$path_k(n_i) = (path_{k-1}(n_i) \cup BH(n_k)) - n_{k+1}. \tag{2.23}$$

In Eq. 2.23, n_k is the current node during graph traversal (the first n_k is n_i and $path_{k-1}(n_i)$ is $path_0(n_i)$) and n_{k+1} is the next node to traverse in the graph which has the closest temporal relation with n_k (current node), and exists in $path_{k-1}(n_i)$ or $BH(n_k)$. Recall that $BH(n_k)$ contains one or more parent node. Traversing graph upward and updating *path* set will be continued until we reach a root node in the graph (in the root node the *path* set will be again empty and this means that we have stop traversing the graph). So the split blob would be labeled as the same as blob related in the root node that we reached in hypothesis graph. A split blob which has an empty *BH* set would be identified as a new identified object with new label; this means this node is a root node.

2.2.6 Finding trajectory

When an object leaves the scene, its trajectory is constructed (step 4, figure 2.3). To do this, first the hypothesis graph is traversed upward starting from hypothesis node

belonging to leaving blob as explained in section 2.2.5. During graph traversal, each node which is deleted from *path* set it is added to another set which is named *Trajectory* set (because this node is the next node to traverse in the graph). When the current node is a root node, we have the *Trajectory* set. But in the hypothesis graph some part of the trajectory where the object was tracked in a group is missing; because group blobs have no nodes in hypothesis graph (figure 2.10 at right). So the missing part of the *Trajectory* set is recovered by searching in the event graph (figure 2.10 at left).

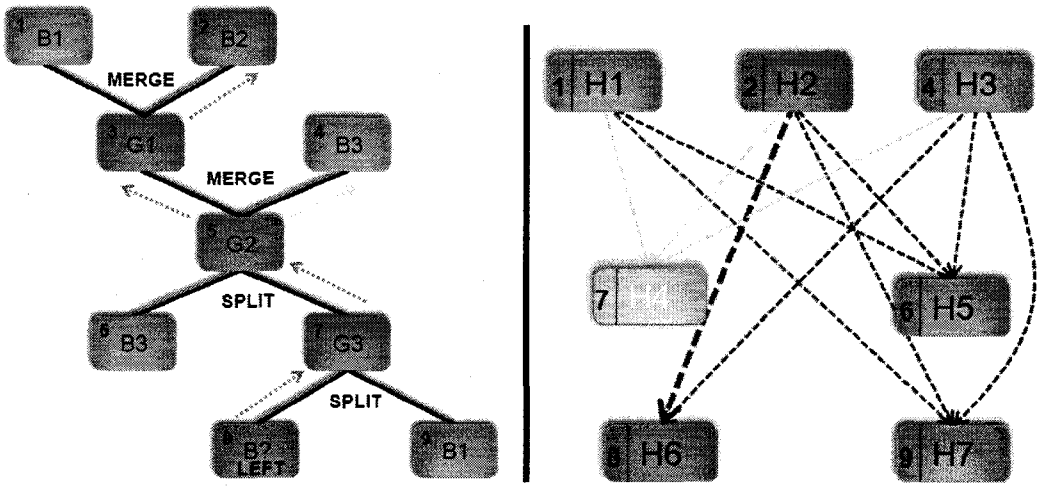

Figure 2.10 Finding trajectory (left image Event graph and right image hypothesis graph). In the event graph, the number in the upper left corner of each node is a pointer to a hypothesis node (number at the left).

Figure 2.10 shows an example for finding trajectory. In this figure, each track node of the event graph has an index i in the top left corner of node. So this node will be called n_i. In the event graph, the blob n_8 has left. To find its trajectory for its corresponding hypothesis node (H_6) in hypothesis graph, the *Trajectory* set is defined as $Trajectory(n_6) = \{n_6, n_2\}$. The corresponding track node for H_6 in the hypothesis graph is n_8 in the event graph and for H_2 in the hypothesis graph it is n_2 in the event graph. To find the missing part of the trajectory between the two nodes n_8 and n_2 in the event graph, searching is done upward in the graph starting from n_8. The goal of the event graph traversal is to find n_2, so we search recursively in the graph to find n_2. When n_2 is

found the path in the graph between n_8 to n_2 is added to *Trajectory* set. Since n_2 is a root node in the event graph the trajectory is completed.

CHAPTER 3 RESULTS AND DISCUSSION

In this chapter, the results of our tracking algorithm are presented and validated by some performance metrics which will be described later in this section. Our tracking system is programmed in the Visual C++ environment utilizing *The Open Computer Vision Library* (OpenCV is a collection of algorithms and sample codes for various computer vision problems) [52]. A computer configured with dual 3.4 GHz Intel Xeon(tm) is used for obtaining our tracking results.

The input of our tracking system is image frames of a video sequence after performing background subtraction. The outputs of our tracking system are: 1) object labeling in each time frame during the processing of the video sequence 2) trajectory of each tracking target in the video sequence.

Background subtraction (BS) is used as preprocessing of our tracking system. We utilize two BS methods: Temporal average [2] and RectGauss (A combination of block-based and Gaussian mixture methods), which has been developed in our laboratory (LITIV). Depending on the video sequence, we used one of these two BS methods as the preprocessing of our tracking system. The BS methods also contain noise filter and shadow detector which eliminate very small detected blobs (like noises) and shadows of objects.

We did not compare our results with other works, because we did not have access to the background subtraction and parameters that they used as pre-processing of the tracking. Instead, we analyzed our tracking algorithm with some metrics to show the capabilities of our tracking system in some complex tracking scenarios.

The video sequences that we used in our experiments are from CAVIAR Datasets and LITIV laboratory video sequences which are described as follows:

- *CAVIAR Test Case Scenarios* [53]: CAVIAR dataset is a number of video clips with different scenarios of interest. These include people walking alone, meeting

with others, window shopping, entering and exiting shops, fighting and passing. First set of videos in this dataset are captured with a wide angle camera lens in the entrance lobby of the INRIA Labs at Grenoble, France. The resolution is half-resolution PAL standard (384 x 288 pixels, 25 frames per second). The second set of videos is also captured with a wide angle lens camera along and across the hallway of a shopping centre in Lisbon. The resolution is half-resolution PAL standard (384×288 pixels, 25 frames per second).

- *LITIV Laboratory Test Case Scenarios*: LITIV dataset are home made video sequences in three different tracking environments. The video sequences are captured in LITIV laboratory, in atrium of Lassonde building, and finally in the parking outside of Lassonde building. The image size of videos is 800 × 600 captured by a Sony DFW-SX 910 at 7.5 fps.

The reminder of this chapter is organized as follow, in section 3.1 the methodology of our experiments are described and in section 3.2 the test case scenarios with their discussion are presented.

3.1 Methodology of experiments

In this section, the methodology for the validation of the tracking results is described. As we mentioned earlier, our tracking algorithm has two types of outputs, first object identification in each frame by labels (online output) and second object trajectories at the end of video sequence (offline output). For validation of the online output (object labeling), we used qualitative observation by a human operator; this means the labels of objects are compared with the interpretation of a human operator while the tracking system is executing. And for object trajectories, we compare the computed trajectories with the ground-truth trajectories that we generated semi-automatically. Generation of ground-truth trajectories and the performance metrics for online labeling and computed trajectories are described in details in next subsections.

3.1.1 Generating ground-truth trajectories

The goal of generating ground-truth trajectories is having a basis to evaluate the accuracy of trajectories constructed by our tracking system for tracking targets in the video sequence. Before describing the generation of ground-truth trajectory, we should define a trajectory. A trajectory of an object is defined as the temporal sequences of the blob's centroid position in the image for the period of time that it exists in the FOV of camera. It is also important to note that the centroid location of a tracked object while it is interacting with other objects (in a group) is the centroid of group blob and not the centroid of individual object. The centroid's location of an object is affected by two factors which are the object detection method and the tracking system. Since we are only interested in estimating the influence of our tracking system on object trajectory and not the influence of object detection method, to generate our ground-truth, we used the same background subtraction method as for our tracking system.

The semi-automatic interactive ground-truth generator consists of: 1) automatic background subtraction 2) automatic blob centroid location 3) Ground-truth Marking Tool (GTM). The background subtraction part is used to determine the foreground blobs at each frame and is the same that we used as the preprocessing of our tracking system.

The blob centroid location detects the location of blob centroids at each frame. And finally, the GTM is an interactive tool that enables the user to click with the mouse on the desired centroid point on the image frame to construct the trajectories. We utilize *cvSetMouseCallback()* function from OpenCV library which has the ability to assign a callback for mouse events such as left click or right click.

In figure 3.1, the left image is the video sequence frame and the right image is the frame image after applying the ground-truth generator. In the right image, the circles on blobs represent the location of blob centroid and the (+) is the mouse pointer that the user can use to click on one of the centroid points. User can generate the ground-truth trajectory, frame by frame by clicking on a desired centroid point of object.

Figure 3.1 Generating ground-truth

3.1.2 Performance metrics

To evaluate the performance of online labeling a metric is defined as follow:

- *Purity of object labeling:* this measurement is how often a blob containing an identified object (identified object is an object which is known as an independent

entity) contains the label of that object in its labels during tracking. Purity is defined as

$$P(O_i) = \frac{N_{gl}}{N_R} \tag{3.1}$$

In Eq. 3.1, N_{gl} (Number of good labels) is the number of frames that the label of identified object O_i is included in the labels of the blob containing that object. N_R is the number of frames that object exist in the FOV of camera. The maximum value for this measurement is 1 which means object's label is always correct during tracking of the object.

To evaluate the accuracy of computed trajectories, first a ground-truth trajectory sequence is compared with trajectory results to find the best matches which reduce a defined distance measure (Eq. 3.2) between two trajectories. Then, some metrics are defined to evaluate different aspects of the computed trajectory compared to its matched ground-truth trajectory. It is important to mention that the small blobs which are in the scene for only a very short time are assumed as noises and ignored in both generating ground-truth trajectories and also computed trajectories.

The distance measure and metrics are the ones used in [1]. The distance measure for comparing a computed trajectory with a ground truth trajectory is defined as follows

$$D(T_c, T_G) = \frac{1}{N_{C,G}} \sum_{i:\exists T_C(t_i)\&T_G(t_i)} \sqrt{(x_C(i) - x_G(i))^2 - (y_C(i) - y_G(i))^2} \,. \tag{3.2}$$

In Eq. 3.1, T_C is the computed trajectory sequence, T_G is the ground-truth trajectory sequence $N_{C,G}$ is the number frames in both T_C and T_G trajectories (the existing frames in computed and ground-truth trajectory are not always the same), and finally $(x_C(i), y_C(i))$ is a point in the computed trajectory at frame t_i and similarly

$(x_G(i), y_G(i))$ is a point in the ground-truth trajectory at frame t_i. The performance metrics are defined as

- *Centroid location error* [1]: this measurement is computed by the distance between the ground-truth trajectory and the computed trajectory of our tracking system. The distance is the one defined in equation 3.1. This measure is useful to determine how close is the computed trajectory to ground-truth trajectory for frames existing in both trajectory sequences. This measurement reveals the errors of approximating the centroid of object while there is occlusion or fragmentation. The unit of this measurement is pixel. And smaller values for this measurement mean the trajectory is more accurate.

- *Track incompleteness factor* [1]: this factor measures how well the computed trajectory covers the existing frames in the ground truth. It is defined as

$$TI = \frac{F_{nf} + F_{pf}}{N_{C,G}}. \tag{3.3}$$

  In Eq. 3.2, F_{nf} is the false negative frame count (the number of frames that are missing from the computed trajectory), F_{pf} is the false positive frame count (the number of frames that are existed in computed trajectory but not in the ground-truth trajectory, and $N_{C,G}$ is the number of frames present in both computed and ground-truth trajectories (intersection frames). Smaller values for this measurement mean computed trajectory better covers the frames existing in ground-truth trajectory. The unit of this factor is frame.

- *Track error rates* [1]: this error measures the false positive rate f_p and the false negative rate f_n as ratios of number of ground truth trajectory frames. It is defined as

$$f_p = \frac{F_p}{N_{GT}} \text{ and} \tag{3.4}$$

$$f_n = \frac{F_n}{N_{GT}}. \tag{3.5}$$

In Eq. 3.3, F_p is the number of computed trajectories without corresponding ground-truth trajectory, and in Eq. 3.4 F_n is the number of ground-truth trajectories without corresponding computed trajectory, and N_{GT} is the number of ground-truth trajectories. For F_p and F_n, smaller values mean smaller error and their unit is frame.

- *Average position error* [1]: this error is the average of all the existing position errors belonging to computed trajectories. The unit of this error is pixel
- *Average track incompleteness* [1]: this error is the average track incompleteness error of all computed trajectories. The unit of this factor is frame.

The above metrics are use to evaluate the final trajectory results of our tracking system. It is important to mention that since our tracking algorithm is adaptive, it may be possible to correct wrong identifications in later frames. Hence, the results of object labeling are not always the same as final computed trajectory for the object.

3.2 Test case scenarios

In this section, several video test cases are chosen to show different aspects of our algorithm. In these test cases, the advantages and limitations of our tracking system are investigated. The video sequences are obtained from LITIV videos and CAVIAR video dataset. The scenarios of the test cases that we have chosen are: 1) outdoor 2) left-baggage 3) label correction 4) Fragmented object 5) Crowd. The outdoor scenario tests the ability of our system to handle scenes with lots of shadows and waving trees, which cause poorer background subtraction and more fragmentation. The left-baggage scenario tests the ability that how correct our system can reconstruct the trajectory of new independent identified objects. The scenario related to label correction tests the ability of our system to correct the wrongfully identified objects in later frames. The scenario

related to fragmented object reveal the ability of the fragmentation checking module of our system to distinguish between fragmentation and splitting. Finally the crowd scenario tests the ability of our system for multiple object tracking.

3.2.1 Test Case 1 (Tracking outdoor)

This test case is about tracking objects outdoor. The two videos are chosen from LITIV video sequences. They are captured outside of pavillon Lassonde, École polytechnique. For these videos, we used the *RectGauss* BS as preprocessing. The first scenario is in the parking and the second is on the stairs. The frame size of processing for both videos is 480×360. The frame size of original video is 800×600, but they are resized to 480×360 because the parameters values used for background subtraction are chosen for the reduced size which gives the minimum errors in background subtraction.

3.2.1.1 Outdoor (Parking)

In the parking video sequence, three people come together in the scene then in later frames, they separate from each other. At last, each one leaves the scene individually. This video sequence has 368 frames. Figure 3.2 shows parking video sequence.

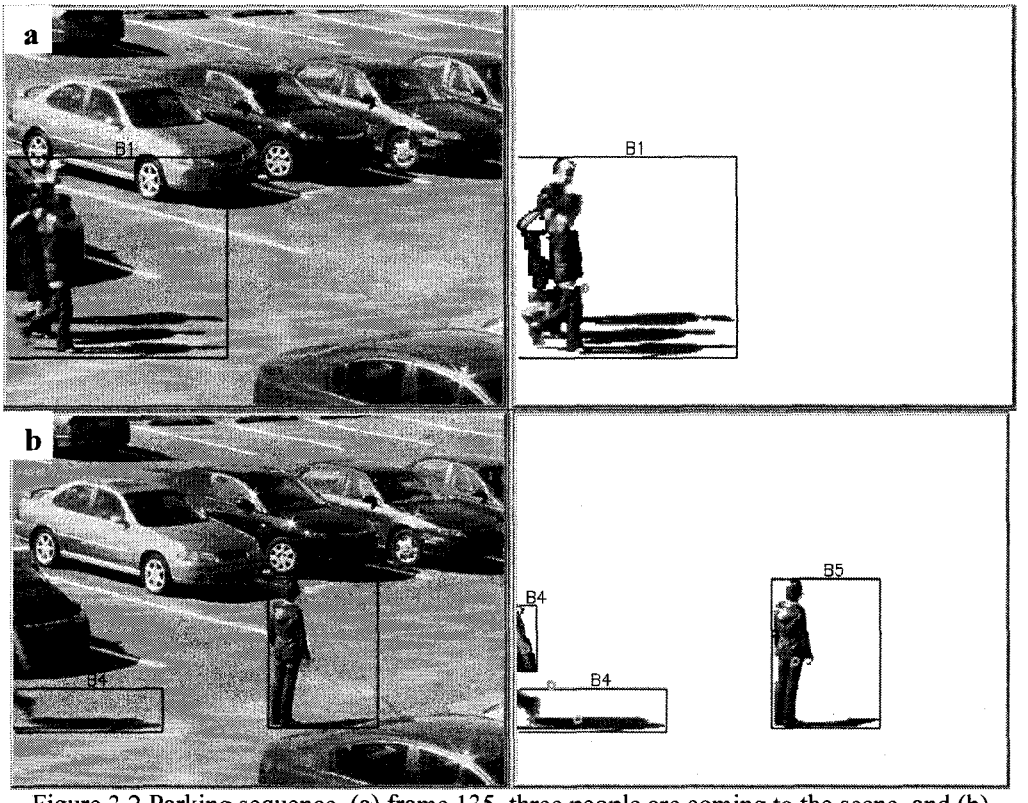

Figure 3.2 Parking sequence. (a) frame 135, three people are coming to the scene, and (b) frame 183, the first person has left the scene and the second one is leaving the scene while the third one remains in the scene.

It can be noted from figure 3.2 that pixels related to shadows are connected to blob region. So shadows reduce the accuracy of blob centroid detection. Figure 3.2 (a) shows three people enter to the FOV of camera as a group. Our algorithm is able to detect each person as an independent moving object when it is separated from the group. The group blob in figure 3.2 (a) is labeled as B1 and in the time that independent objects are detected, they took new labels as *B2*, *B3*, and *B4*. The problem that shadow causes in the level of tracking is that for second and third object, the shadow is separated and it is leaving the scene later than objects; so it detected as a new object which is a false positive result. You can see this situation in figure 3.2 (b) that *B4* is fragmented to two blobs and in the frame after this one (the one which is shown) *B4* leave the scene while its shadow is still in the scene. Table 3.1(a) shows trajectory error for each object, and table 3.1(b) the total computed trajectories errors.

Table 3.1 a) trajectory errors for each object in parking sequence, the object ID# are the IDs that individual blob take during its track, for example ID# 1, 2 means first object was in *B1* and then in *B2* b) total computed trajectories errors.

a)

Object ID#	D	F_{nf}	F_{pf}	$N_{R,G}$	TI	p
1,2	2.46521	0	0	56	0	1
1,4	1.81954	0	0	60	0	1
1,5	0.509406	0	0	162	0	1

b)

Test case#	**1**
Track error f_p	2/3
Track error f_n	0
Average position error	1.59805
Average track incompleteness	0

In table 3.1 (a) the centroid position error (D) is because objects are separated and again merged within 4 frames before fragmentation checking module can distinguish the splitting and fragmentation. So they are assumed as fragmentation. In this case there is a small centroid position error. Also, it can be noted from table 3.1 (a), that exists no *TI* (Track Incompleteness error) and *P* (purity) is 1 for all, which means three objects are correctly identified and tracked in all the frames that are in the FOV of camera. Table 3.1 (b) shows tracking system detected two more objects compare to ground-truth which are false positive error; this is because of shadows detected as individual object.

3.2.1.2 Outdoor(Stairs)

In the stairs video sequence, one people enter the scene and put a bag on the ground and then leave the scene. The other person goes down the stairs and pick-up the bag and leaves the scene. Finally two people enter to the scene, exchange a bag and then leave the scene. This video sequence has 569 frames. Figure 3.3 shows three frames from the stair video sequence.

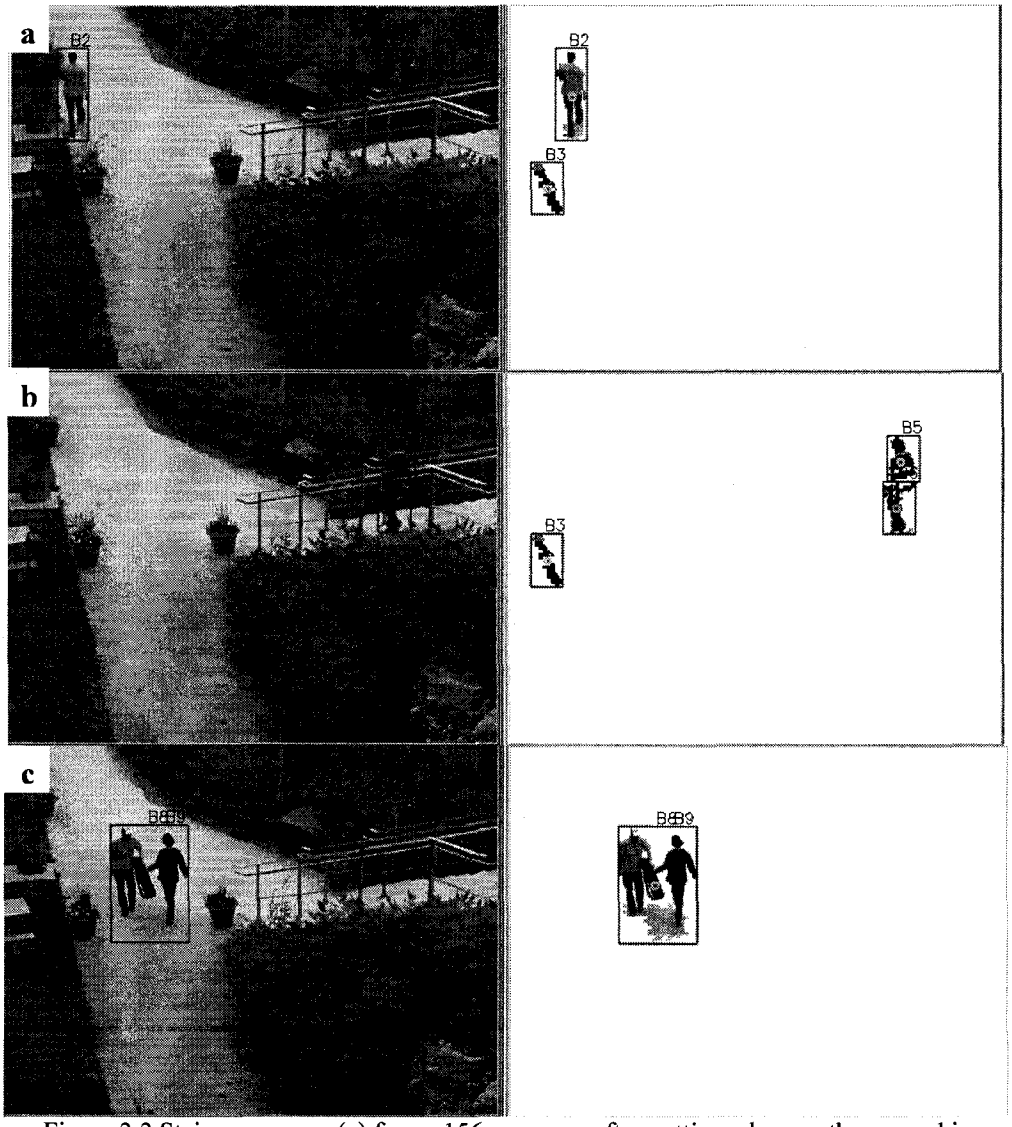

Figure 3.3 Stairs sequence. (a) frame 156, a person after putting a bag on the ground is leaving the scene, (b) frame 218, another person is going down the stairs to pick up the bag and leave the scene, and (c) frame 510, two people is exchanging a bag and leaving the scene.

It can be noted from figure 3.3 (a), tracking system can correctly identify the bag as an independent object. Figure 3.3 (b) shows the railing of stairs caused the fragmentation, which is detected wrongfully as splitting (blobs *B5* and *B6*). This fragmentation is repeated in several consecutive frames while the number of fragments changes a lot. This consecutive fragmentation results to the detection of two false positive independent objects as *B6* and *B7* in the fragments of the blob that is going down the stairs (*B5*). Figure 3.3 (c) shows when people renter the scene, they are detected as new objects with new labels and when they exchange the bag, bag is not an independent entity so it does not take a separate label. Table 3.2(a) shows trajectory error for each object, and table 3.2(b), the total computed trajectories errors.

Table 3.2 a) trajectory errors for each object in stairs sequence b) total computed trajectories errors

a)

Object ID#	D	F_{nf}	F_{pf}	$N_{R,G}$	TI	p
1,2	0.987586	0	0	131	0	1
1,3	0.499955	0	0	246	0	1
5	0.591913	0	0	110	0	1
8	0.207495	0	0	145	0	1
9	0.216932	0	0	135	0	1

b)

Test case#	**1**
Track error f_p	2/5
Track error f_n	0
Average position error	0.500776
Average track incompleteness	0

In table 3.2 (a), the position error (D) is because of fragmented object that went down the stairs and also the error of the person who generates the ground-truth with mouse click (this means the mouse is clicked on the centroid with small error). It can be noted from table 3.2(a); all the existing objects are correctly identified and tracked in the frames that they existed in FOV of camera. In table 3.2 (b) two false positive trajectories are because of two wrongfully detected objects resulting from severe fragmentation because of the railing.

This test case showed our algorithm is able to track objects correctly without loosing its trajectory. Also it is able to identify new blob entities from the first frame in the video in which it is moving independently (example: the bag). But still, our algorithm suffers from the limitation of BS module which causes fragmentation that may results to detect false positive objects and shadows that may results to non-accurate blob centroid position.

3.2.2 Test Case 2 (Left-baggage)

This test case is about identifying and tracking left baggage in the scene. The two videos are chosen in order from CAVIAR dataset and LITIV video sequences. The video from CAVIAR dataset is in a hallway. For this video t*emporal average* BS is used as preprocessing and its frame size for processing is 384×288. For video from LITIV in the parking outside the Lassonde building, *RectGauss* BS is used as preprocessing and its frame size for processing is 480×360.

3.2.2.1 Left-baggage(Hallway)

In the Hallway video sequence, a person enters the scene with a bag then puts the bag on the floor and stays stationary near the bag for a long period of time. Then he goes far from the bag and at last he comes back and takes the bag and leaves the scene. This video sequence has 1226 frames. Figure 3.4 shows hallway video sequence.

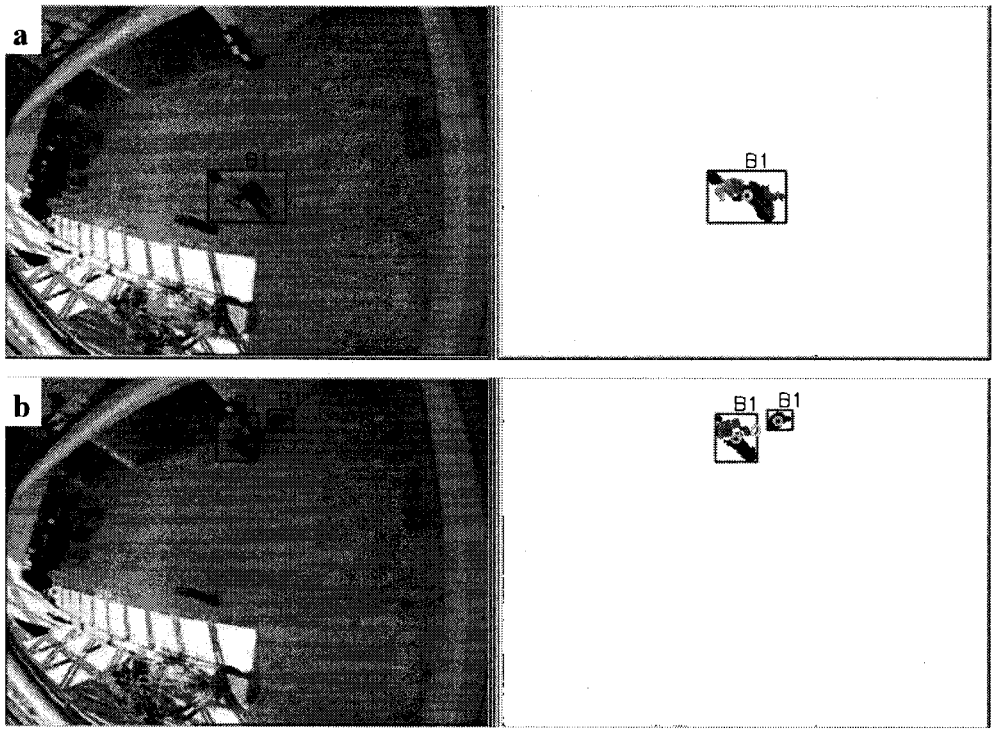

Figure 3.4 Hallway sequence (a) frame 216, a person is entering the scene with a bag and (b) frame 412, same person is putting bag on the floor and staying near the bag [53].

Figure 3.4 (b) shows the situation where a person puts a bag on the floor. Consecutive fragmentation and remerging happen because the pants of the person and the shelf in the background are black. At last the person stays stationary near the bag. While the person is remaining stationary, fragmentation checking module could not detect the bag as an independent entity, and the bag is assumed as a fragment of person. But when the person goes far from the bag, bag is detected as a new object. Table 3.3(a) shows trajectory error for each object, and table 3.3(b), the total computed trajectories errors.

Table 3.3 a) trajectory errors for each object in hallway sequence b) total computed trajectories errors

a)

Object ID#	D	F_{nf}	F_{pf}	$N_{R,G}$	TI	p
1,3	11.0981	0	0	868	0	0.6255
1,2	33.6376	0	0	868	0	0.6255

b)

Test case#	**2**
Track error f_p	0
Track error f_n	0
Average position error	22.3678
Average track incompleteness	0

As it is shown in the table 3.3 (a), centroid positions errors (D) exists for both objects (bag and person). It is because during the time that person and bag are still assumed as fragments their centroid is the average centroid of both objects. Also their label is *B1* which is not a good label; so the purity factor (p) of object is less than one. In table 3.3 (a) all the *TI*s are zero, which means objects are tracked correctly in all frames and their final computed trajectory is correct. You can also observe in table 3.3 (b) the average position errors which is large because of the reason just explained.

3.2.2.2 Left-baggage(parking)

In the parking video sequence, two people enter the scene. After, they merge and separate from each other. Then, one of them leaves a bag in the scene and also a car passes on the road. This video sequence has 244 frames. Figure 3.5 shows parking video sequence.

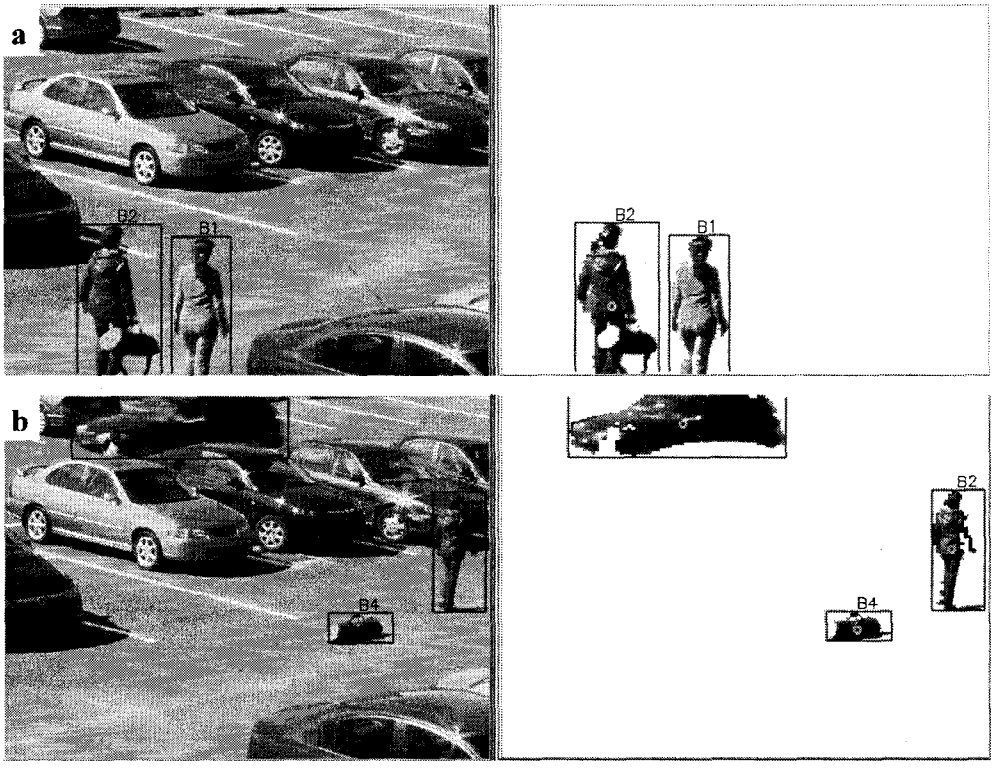

Figure 3.5 parking sequence (a) frame 62, two persons which one of them carries a bag enter to the scene and (b) frame 181, the person has left the bag and also a car is passing.

It is important to notice that the person carrying the bag had interaction with the person which has left the scene before the bag is identified as an independent object. During that time, the trajectory of the bag does not exist, because its presence is unknown before it is split as an individual object. So we can only know that it existed in the last group it was part of, and not any previous group. Table 3.4(a) shows trajectory error for each object, and table 3.4(b), the total computed trajectories errors.

Table 3.4 a) trajectory errors for each object in parking sequence b) total computed trajectories errors

a)

Object ID#	D	F_{nf}	F_{pf}	$N_{R,G}$	TI	p
1	6.47861	0	0	109	0	1
2	4.96091	0	0	142	0	1
3	0.290893	0	0	19	0	1
2,4	3.89144	19	0	180	0.10555	0.93888

b)

Test case#	**2**
Track error f_p	0
Track error f_n	0
Average position error	11.7152
Average track incompleteness	0

In table 3.4 (a), 19 false negative frames exist as trajectory error for the bag, because for 19 frames, the bag was not detected as independent object. The error on centroid positions are because of the consecutive fragmentation and remerging of objects and shadows. Eleven (11) frames error of labeling (p) for bag is because of consecutive fragmentation after splitting which postpone the identification of splitting and consecutively caused delayed object labeling. As it is shown in table 3.4 (a), all other objects are labeled and tracked correctly for all the frames that they are in the FOV of camera.

This test case shows our algorithm has delay on online labeling because of consecutive fragmentation at the same time that splitting happens. Our tracking system does not lose

tracking of those frames in the final computed trajectory. So, it only affects the online labeling, not the final trajectory. Objects are tracked but with centroid error position for those frames.

3.2.3 Test Case 3 (label correction)

This test case is about consecutive merging and splitting and shows how our algorithm corrects the wrong labeling. Video is chosen from LITIV video sequences and it is captured in Atrium of pavilion Lassonde. For this video, we used *RectGauss* BS as preprocessing and its frame size for processing is 480×360. Total number of frames of video is 232. The video quality is low. In this video, two persons enter to the scene while several interactions happen between them. During this time, two other persons enter and leave the scene while one of them has interaction with first two persons in the scene. Figure 3.6 shows atrium video sequence.

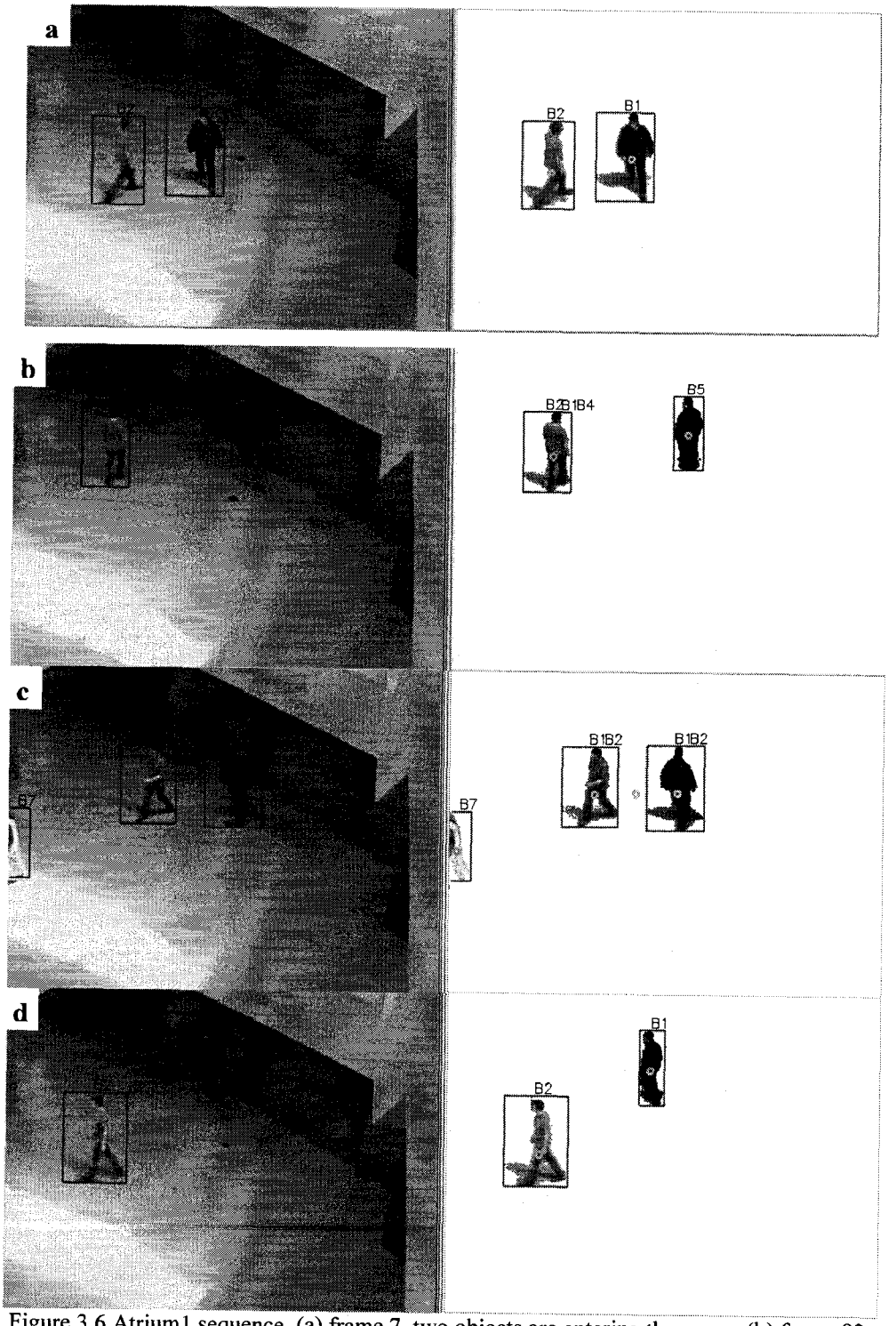

Figure 3.6 Atrium1 sequence, (a) frame 7, two objects are entering the scene, (b) frame 93, consecutive merging splitting is happening, (c) frame 225, a person is leaving the scene and, (d) frame 230, two persons are leaving the scene

Figure 3.6(b) shows that after consecutive merging and splitting, *B1* and *B2* lose two times their Ids and get two other labels (*B4* and *B5).* They are detected as new objects because the quality of video is low and also there exists a lightening change in the scene which causes wrongful matching. In the later frames *B1* and *B2* are again correctly identified after they have an interaction with *B7* (figure 3.6 (c)). Finally in figure 3.6 (d), *B1* and *B2* are correctly identified. Table 3.5(a) shows trajectory error for each object, and table 3.5(b), the total computed trajectories errors.

Table 3.5 a) trajectory errors for each object in Atrium1 sequence b) total computed trajectories errors

a)

Object ID#	D	F_{nf}	F_{pf}	$N_{R,G}$	TI	p
1	6.47459	0	0	189	0	0.94708
2	4.24021	36	0	156	0.23076	0.96794
3	0	0	0	19	0	1
4	0	0	0	21	0	1

b)

Test case#	**3**
Track error f_p	1
Track error f_n	0
Average position error	5.3574
Average track incompleteness	0.05769

As it can be noted from table 3.5 (a), *B2* loses its track for 36 frames and this 36 frames does not exist in its trajectory. This is because during those frames, false positive labels are detected. For one of them, the trajectory of object is lost and in later frame it is

again identified. In later frames, the algorithm could not correct the trajectory of those lost frames. So that part of the trajectory is detected as a new trajectory which is separated and it belongs to one false positive object. *B1* and *B2* during few frames were not correctly labeled which you can see with their purity factor in table 3.5 (a). As it is shown in table 3.5 (b) a false positive object exists for the wrongfully detected object. The other objects are correctly labeled and tracked.

This test case shows that when our algorithm loses the identification of objects during tracking, by using the information of later frames and relating them to previous observation, it is capable to re-identify the objects and reconstruct the trajectories.

3.2.4 Test Case 4 (Fragmentation)

This test case is about object fragmentation during tracking and shows how our algorithm works for the fragmentation cases. Video is chosen from LITIV video sequences and it is captured in one of the LITIV laboratories. For this video, frame size for preprocessing (*temporal average* BS is used) and processing is 320×240. This is because in this size, we had good result for background subtraction. Total frame numbers of video are 201. In this video, two people enter the scene and occlude each other while there is fragmentation because similar color of moving object with background. Our tracking system correctly distinguishes between fragmentation and splitting and labels objects correctly. Figure 3.7 shows laboratory video sequence.

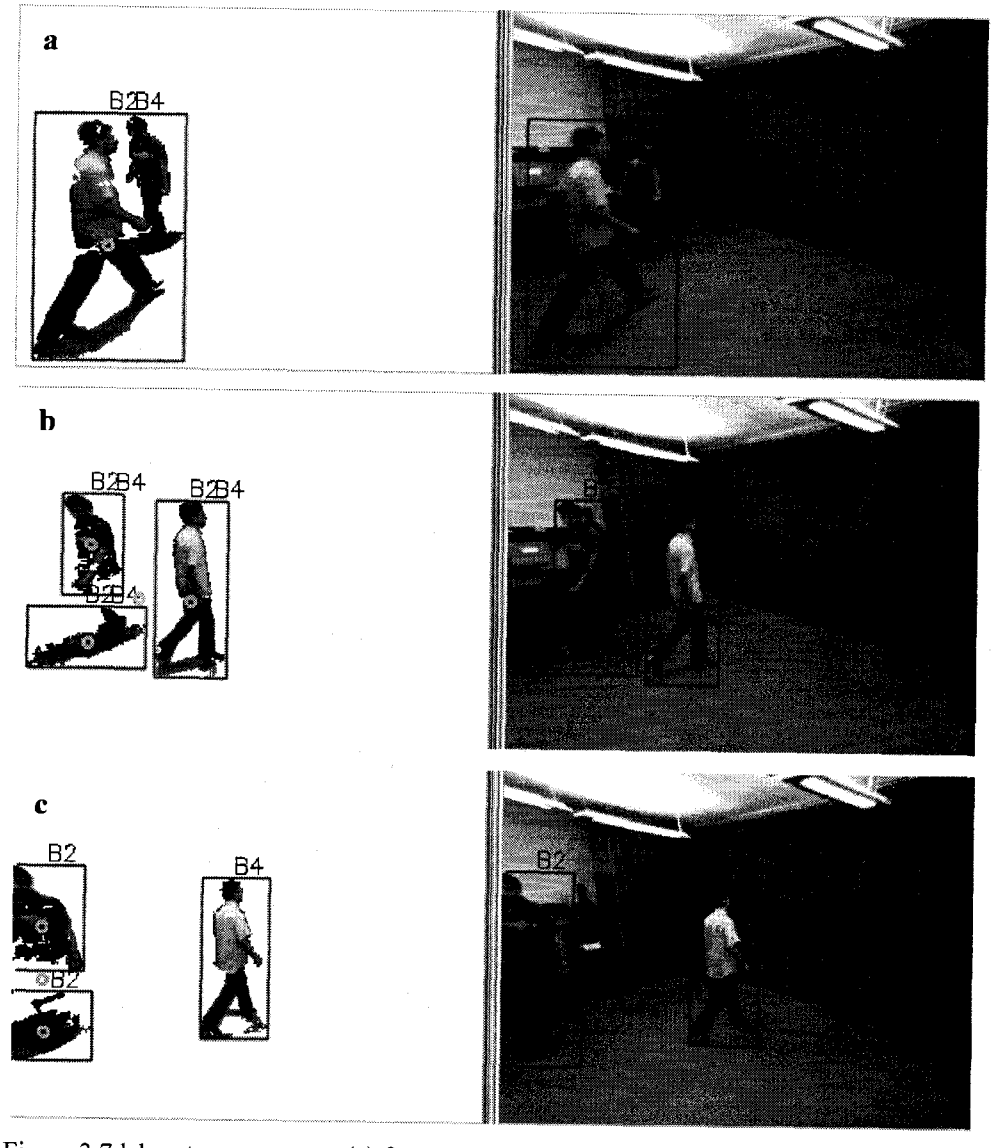

Figure 3.7 laboratory sequence, (a) frame 113, two persons have interaction, (b) frame 119, the persons are splitting while there is a fragmentation, and (c) frame 123, split persons with correct labels.

As it is shown in figure 3.7(c), two persons are correctly labeled as *B2* and *B4* after splitting while the fragmentation remains. Table 3.6(a) shows trajectory error for each object, and table 3.6(b), the total computed trajectories errors.

Table 3.6 a) trajectory errors for each object in laboratory sequence b) total computed trajectories errors

a)

Object ID#	D	F_{nf}	F_{pf}	$N_{R,G}$	TI	p
2	1.28756	0	0	87	0	1
4	0.0915484	0	0	94	0	1

b)

Test case#	**4**
Track error f_p	0
Track error f_n	0
Average position error	0.68955
Average track incompleteness	0

Table 3.6 shows the results for this sequence are correct. And there exists a small error of centroid positions which is caused during generating ground-truth (small error average 1 pixel, when mouse is clicked on centroid position during generating ground-truth).

This test case showed our fragmentation checking is capable to distinguish between splitting and fragmentation.

3.2.5 Test Case 5 (Crowd)

This test case is about tracking in crowd when several people exist in the FOV of camera. For this test case, three video are chosen from CAVIAR dataset and LITIV video

sequences. The video from CAVIAR dataset has been captured in a shopping center. For this video sequence, we used *RectGauss* BS as preprocessing and its frame size for processing is 384×288. The video from LITIV video sequence captured in Lassonde building atrium, for this video sequence, we used *RectGauss* BS as preprocessing and its frame size for processing is 480×360.

3.2.5.1 Crowd(Atrium)

In the Atrium1 video sequence, four people enter the scene, and then they have interactions with each other. Finally, they leave the scene individually. This video sequence has 166 frames. Figure 3.8 shows atrium1 video sequence.

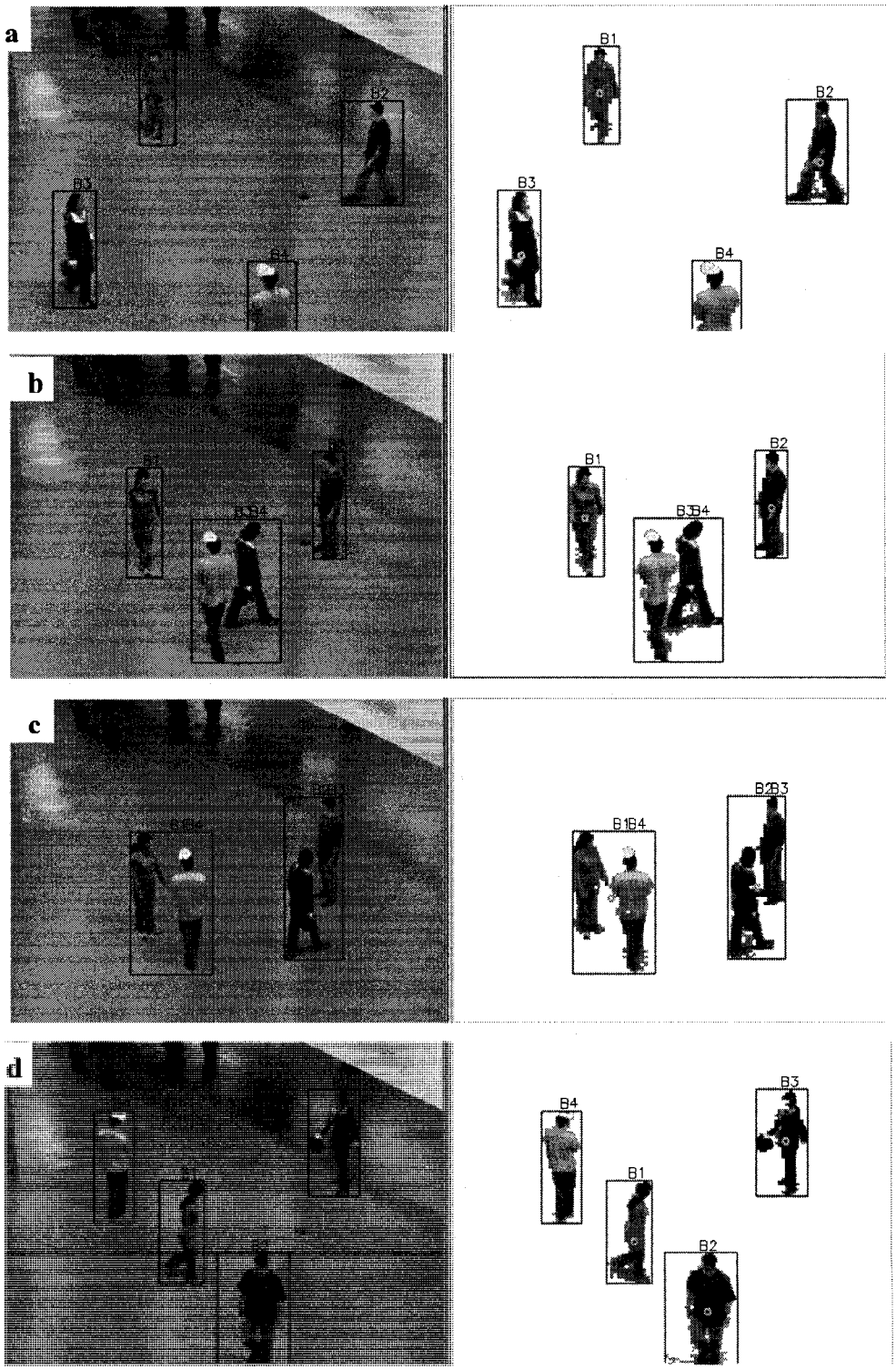

Figure 3.8 Atrium1 sequence, (a) frame 40, four persons are entering the scene, (b) frame 56, an interaction between two objects is happening, (c) frame 65, two other interactions between two pairs of objects are happening, and (d) frame 103, they are leaving the scene

Figure 3.8 (a) shows four persons entering the scene. Figure 3.8 (b) and (c) shows their consecutive interactions. And finally, figure 3.8 (d) shows they are leaving the scene while they are correctly identified. This part of Atrium video sequence is one of the best results that we had and shows the ability of our algorithm for multiple object tracking. Table 3.7(a) shows trajectory error for each object, and table 3.7(b) the total computed trajectories errors.

Table 3.7 a) trajectory errors for each object in Atrium1 sequence b) total computed trajectories errors

a)

Object ID#	D	F_{nf}	F_{pf}	$N_{R,G}$	TI	p
1	0	0	0	111	0	1
2	0	0	0	87	0	1
3	0	0	0	133	0	1
4	0	0	0	101	0	1

b)

Test case#	**5**
Track error f_p	0
Track error f_n	0
Average position error	0
Average track incompleteness	0

The values in table 3.7 (a) and (b) shows there is no trajectory error for this video sequence.

The other part of this video sequence (Atrium 2) reveals one of the possible few errors in our tracking system. Figure 3.9 shows Atrium2 video sequence.

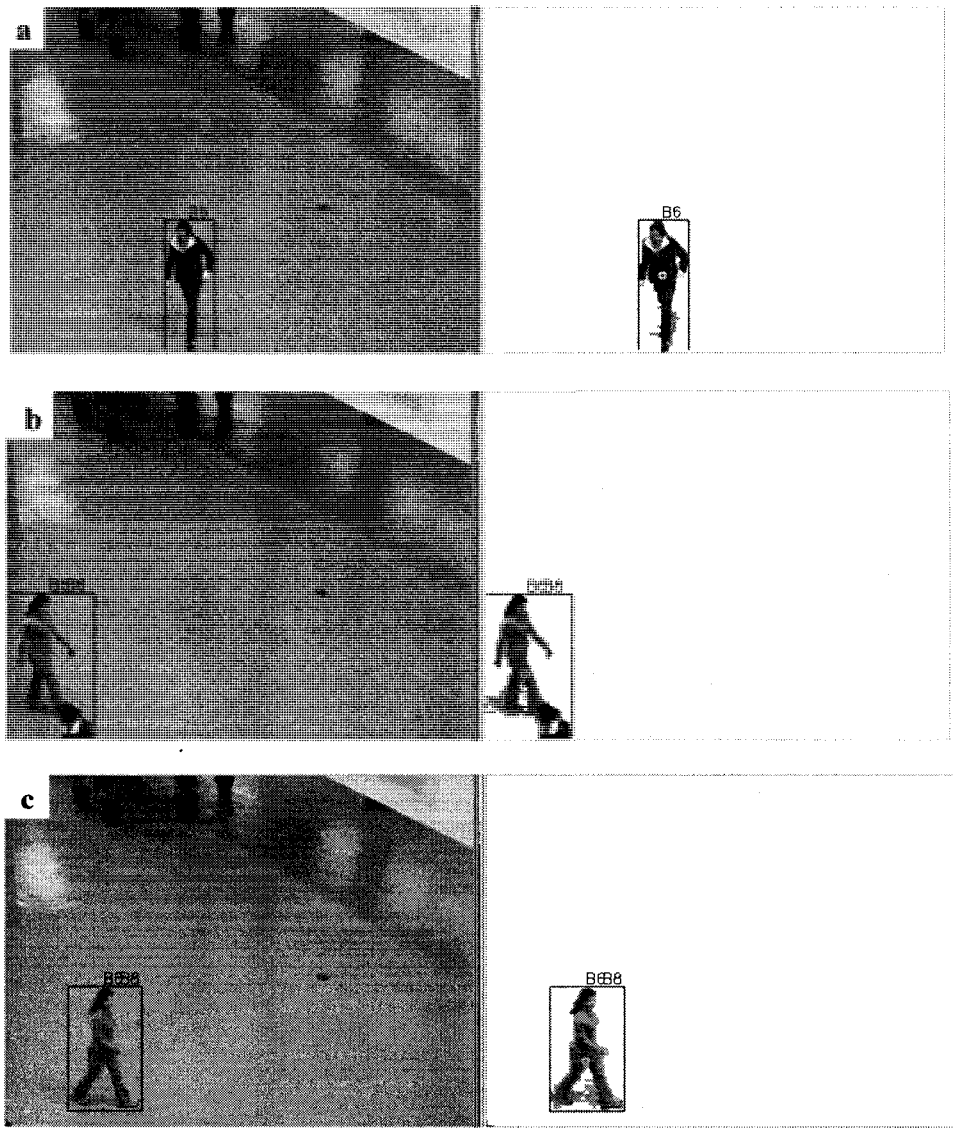

Figure 3.9 Atrium2 sequence, (a) frame 337, a person is leaving the scene, (b) frame 357, another person is entering the scene while it is merging with leaving object, and (c) frame 363, the entering person continues walking around the scene

3.9(c) shows object *B6* has left the scene, but its label remains in group blob. This is because in figure 3.9 (b) our tracking system do not know that object *B6* is leaving because of the merging without splitting. So tracking system thinks that the left object is

still in the scene until the second object leaves the scene. Table 3.8(a) shows trajectory error for each object, and table 3.8(b), the total computed trajectories errors.

Table 3.8 a) trajectory errors for each object in Atrium2 sequence b) total computed trajectories errors

a)

Object ID#	D	F_{nf}	F_{pf}	$N_{R,G}$	TI	p
6	0	0	143	98	0.55642	1
8	0	0	0	152	0	1

b)

Test case#	**5**
Track error f_p	0
Track error f_n	0
Average position error	0
Average track incompleteness	0.2782

Table 3.8 (a) shows false positive frames, because the leaving of *B6* is not detected and it is assumed that it existed in the scene until the second object leaves the scene. The other object is correctly identified and tracked.

3.2.5.2 Crowd(Shopping center)

In the shopping center sequence, three people go out of a store and then walk along the corridor altogether. They separate and two of them have an interaction with another person which enters the FOV of camera while others are leaving the scene. This video is a very challenging sequence since there are reflection on the floor and also color similarity between the person with a white shirt and the floor. This video sequence has 421 frames. Figure 3.10 shows shopping center video sequence.

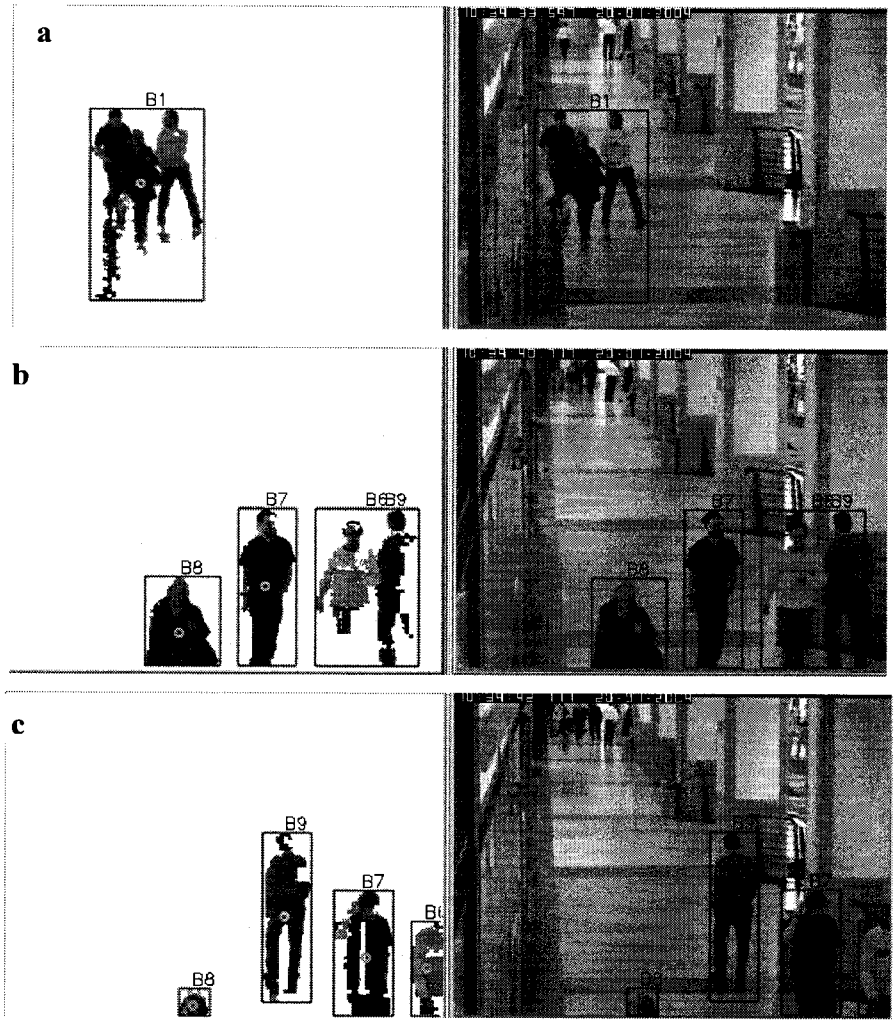

Figure 3.10 shopping center sequence, (a) frame 135, three persons are going out of a store, (b) frame 311, three persons are walking along the corridor while one of them has interaction with a new entering object, and (c) frame 362, three persons are leaving the scene while the fourth one continues walking along the corridor [53].

Table 3.9(a) shows trajectory error for each object, and table 3.9(b), the total computed trajectories errors.

Table 3.9 a) trajectory errors for each object in shopping center sequence b) total computed trajectories errors.

a)

Object ID#	D	F_{nf}	F_{pf}	$N_{R,G}$	TI	p
1, 3,6	5.17013	0	0	275	0	1
1,7	13.52120	0	16	281	0.05693	1
1,8	13.28660	0	0	256	0	1
9	2.34343	0	0	115	0	1

b)

Test case#	**5**
Track error f_p	1/4
Track error f_n	0
Average position error	8.58034
Average track incompleteness	0.01423

Table 3.9 (a) shows that the three first persons have considerable centroid position errors. This is because they walked for a long time together in the same direction and with the same speed along the corridor so the fragmentation checking module can detect the splitting only after some delay. And it results in all three persons being tracked as a group. The other error for object labeled as B1 in the group and B7 as individual is that it came into the scene in a group with two other people, but these two people appeared in the scene 16 frame sooner than third person. So there are 16 false positive frames for that person, because the tracking algorithm can not tell at which moment this individual

object enters the scene as it is first detected in the group. The person with a white shirt had a severe fragmentation because of the similarity between color of floor and his shirt. One of the fragmented parts of the person is detected as a false positive independent object. Then this fragment disappears because of the error of background subtraction. So in the result trajectory, there is one false positive trajectory that you can see in table 3.9(b).

This test case shows the ability of our tracking system for multiple object tracking and also shows the limitation of our algorithm in situation of leaving the scene without splitting (in video sequence atrium 2) and also in situation where an individual object enters in a group (in video sequence shopping center).

CHAPTER 4 CONCLUSION AND FUTURE WORKS

We presented an online tracking system able to track multiple objects. It can handle objects interaction and object fragmentation. Our approach is adaptive, this means it has an online correction module which can correct the errors of objects labeling by exploring the later frames information and relating them to previous observations. Our MOT is also enhanced with a fragmentation checking module that can handle some imperfections of object detection.

The focus of our algorithm is more on introducing a powerful tracking strategy rather than precise object representation. In our experimental results, we measured the performance of our tracking system for online labeling and computed trajectories with appropriate metrics. We showed that using adaptive color histogram (the average color histogram of object in several frames) as the object representation with an adaptive data association allows building a robust online tracking system for realistic tracking scenarios.

Using a graph-based data association method enable us to correct wrongfully labeled objects in later frames by propagating information of previous frames and relating them to information of later frames. Also, the idea of using two graphs, that are built and updated together, and a fragmentation checking module help us to have a more precise hypothesis generation and a reduced search space in the hypothesis graph for data association process.

Contributions of this work

There are some MHT approaches in literature review [6, 10]; our approach can be distinguished from them by the following contributions:

- Our approach for generating hypothesis is not sequential; this means hypotheses are not only generated from related parents one level upper than leaf node, but also from all related parents in upper levels of graph up to root nodes. The

advantage of this approach is that it does not propagate error of data association in one frame to later frames and also recovers the error in a frame by relating information of previous frames to later frames in hypothesis graph. This advantage of our system is shown in test case 3 (section 3.2.3).

- Our system has a fragmentation control module. None of the MHT in literature review account for the errors caused by fragmented objects during the construction of hypothesis graph. The fragmentation control in our algorithm enables us to distinguish between fragmentation and splitting, and initializing hypotheses node and generating hypothesis only for splitting. This results in a reduced size of the hypothesis graph and an accurate data association. This is shown in test case 4 (section 3.2.4).
- We utilized an adaptive color histogram as the measurement for hypothesis generation. The advantage of adaptive color histogram is that it contains global color information about the objects during several frames. This advantage makes that the adaptive color histogram more robust to lighting variations in a short period of time compared to color histogram, built from color information of one frame. Finally this results to a more accurate hypothesis generation. This situation is shown in video sequence of test case 3 (section 3.2.3), in which there is a reflections on a part of the floor. So an object that passes in this region (not staying there for a long time) will have its adaptive color histogram changing a little, but its color histogram will change considerably.

Future works

Our algorithm still suffers from some common problems like other tracking systems. Most of these problems arise in the object detection part (Background subtraction) and lead to erroneous tracking. In other words, the limitations of existing background subtraction methods for video sequences captured by visible camera cause erroneous blob detection in the scene, and this in turn affects the tracking systems.

To improve results and efficiency for our tracking system, we propose the following improvement:

- For object representation, adding texture features to object model may lead to a more accurate object modeling and consecutively more accurate hypothesis generation.
- For very long videos, storing the information of object trajectories in a file and deleting the nodes in the event graph which are deactivated and that will never be reuse until end of video, may lead to a more efficient tracking system.

As a future work in the case of people tracking, in stages of object detection and also object tracking, we propose the following improvements:

- For object detection, the combination of blob detection in video sequences from infrared camera and visible camera may result to more accurate blob detection and consecutively results to better tracking performance. This is because object detection results for infrared video sequence have robustness in similar color structure between background and moving object. On the other hand the object detection results for visible video sequences contain valuable information of blob's color and texture. So the combination of these two may lead to better object detection results.
- For object tracking, the combination of trajectory results from video sequences captured by visible camera and infrared camera may lead to reconstruction of erroneous part of trajectory result for visible video sequences and may lead to better trajectory results.

Tracking consists of low-level event detection such as merging and splitting which are considered as the stage before high-level event detection. High-level event detection can be done by two approaches: 1) accurate object representation to track and interpret different actions 2) study of trajectories patterns. To improve our tracking

system to a high-level event detection system, we can consider the following improvement:

- Defining a more accurate object representation by modeling different part of object and analyzing the motion of those parts for the purpose of action interpretation.
- Defining illegal patterns of object trajectories for specific places (buildings, rooms, lobbies) and give the alarm for those computed trajectories containing illegal patterns.

REFERENCES

[1] A. Senior, A. Hampapur, Y.-L. Tian, L. Brown, S. Pankanti, and R. Bolle, "Appearance models for occlusion handling," *Image and Vision Computing,* vol. 24, pp. 1233-1243, 2006.

[2] B. Shoushtarian and H. E. Bez, "A practical adaptive approach for dynamic background subtraction using an invariant colour model and object tracking " *Pattern Recogn. Lett.* , vol. 26 pp. 5-26, 2005.

[3] M. Shah, O. Javed, and K. Shafique, "Automated Visual Surveillance in Realistic Scenarios," *Multimedia, IEEE,* vol. 14, pp. 30-39, 2007.

[4] H. Weiming, T. Tieniu, W. Liang, and S. Maybank, "A survey on visual surveillance of object motion and behaviors," *Systems, Man and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on,* vol. 34, pp. 334-352, 2004.

[5] C. Motamed, "Motion detection and tracking using belief indicators for an automatic visual-surveillance system," *Image and Vision Computing,* vol. 24, pp. 1192-1201, 2006.

[6] S. J. McKenna, S. Jabri, Z. Duric, H. Wechsler, and A. Rosenfeld, "Tracking Groups Of People " *Computer Vision and Image Understanding: CVIU,* vol. 80, pp. 42-56, 2000.

[7] M. Balcells, D. DeMenthon, and D. DoermannDOI, "An appearance-based approach for consistent labeling of humans and objects in video," *Pattern Analysis & Applications,* vol. 7, pp. 373-385, 2004.

[8] M. Han, W. Xu, H. Tao, and Y. Gong, "Multi-object trajectory tracking," *Machine Vision and Applications,* vol. 18, pp. 221-232, 2007.

[9] C. Alex Yong Sang, H. Weimin, and L. Liyuan, "Multiple Objects Tracking with Multiple Hypotheses Graph Representation," 2006, pp. 638-641.

[10] D. Reid, "An algorithm for tracking multiple targets," *Automatic Control, IEEE Transactions on,* vol. 24, pp. 843-854, 1979.

[11] A. Yilmaz, O. Javed, and M. Shah, "Object tracking: A survey," *ACM Comput. Surv.* , 2006.

[12] P. Nillius, J. Sullivan, and S. Carlsson, "Multi-Target Tracking - Linking Identities using Bayesian Network Inference," in *Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on*, 2006, pp. 2187-2194.

[13] K. Rerkrai and H. Fillbrandt, "Tracking persons under partial scene occlusion using linear regression," in *ISCEE*, 2004.

[14] B. Bose, X. Wang, and E. Grimson, "Detecting and tracking multiple interacting objects without class-specific models," vol. Massachusetts Institute of Technology Computer Science and Artificial Intelligence Laboratory, 2006.

[15] M. B. Capellades, D. Doermann, D. DeMenthon, and C. Rama, "An appearance based approach for human and object tracking," in *Image Processing, 2003. ICIP 2003. Proceedings. 2003 International Conference on*, 2003, pp. II-85-8 vol.3.

[16] I. Haritaoglu, D. Harwood, and L. S. Davis, "W4: real-time surveillance of people and their activities," *Pattern Analysis and Machine Intelligence, IEEE Transactions on,* vol. 22, pp. 809-830, 2000.

[17] M. S. Allili and D. ZiouDOI, "Active contours for video object tracking using region, boundary and shape information," *Signal, Image and Video Processing,* vol. 1, pp. 101-117, 2007.

[18] T. Komuro and M. Ishikawa, "A Moment-based 3D Object Tracking Algorithm for High-speed Vision," in *Robotics and Automation, 2007 IEEE International Conference on*, 2007, pp. 58-63.

[19] L. Yang and F. Albregtsen, "Fast computation of invariant geometric moments: a new method giving correct results," in *Pattern Recognition, 1994. Vol. 1 - Conference A: Computer Vision & Image Processing., Proceedings of the 12th IAPR International Conference on*, 1994, pp. 201-204 vol.1.

[20] L. G. Shapiro and G. C. Stockman, *Computer vision*, Prentice Hall ed.: Tom Robbins, 2001.

[21] D. Cardani, "Adventures in HSV space," in *The advanced Developers Hands on Conference*, 2001.

[22] R. G. Cucchiara, C.; Piccardi, M.; Prati, A., "Detecting Objects, Shadows and Ghosts in Video Streams by Exploiting Color and Motion Information " in *Proceedings of the 11th International Conference on Image Analysis and Processing* IEEE Computer Society, 2001.

[23] R. Cucchiara, C. Grana, M. Piccardi, and A. Prati, "Detecting moving objects, ghosts, and shadows in video streams," *Pattern Analysis and Machine Intelligence, IEEE Transactions on,* vol. 25, pp. 1337-1342, 2003.

[24] L. Peihua, "A clustering Based Color Model and Fast Algorithm for Object Tracking," in *Pattern Recognition, 2006. ICPR 2006. 18th International Conference on*, 2006, pp. 671-674.

[25] D. H. Z. a. P. D. D. F. Dr. Fuhui Long, "fundamentals of content-based image retrieval," 2003.

[26] R. Bourezak and G. Bilodeau, "Object detection and tracking using iterative division and correlograms," in *Computer and Robot Vision, 2006. The 3rd Canadian Conference on*, 2006, pp. 38-38.

[27] R. C. Gonzalez and R. E. Woods, *Digital image processing*: Prentice Hall, 2002.

[28] V. Takala and M. Pietikainen, "Multi-Object Tracking Using Color, Texture and Motion," in *Computer Vision and Pattern Recognition, 2007. CVPR '07. IEEE Conference on*, 2007, pp. 1-7.

[29] T. Feng and T. Hai, "Non-orthogonal Binary Expansion of Gabor Filters with Applications in Object Tracking," in *Motion and Video Computing, 2007. WMVC '07. IEEE Workshop on*, 2007, pp. 24-24.

[30] D. A. Forsyth and J. Ponce, *Computer vision a modern approach*: Prentice Hall, 2003.

[31] S. Dubuisson, "Recursive Clustering for Multiple Object Tracking," in *Image Processing, 2006 IEEE International Conference on*, 2006, pp. 2805-2808.

[32] Y. Changjiang, R. Duraiswami, and L. Davis, "Fast multiple object tracking via a hierarchical particle filter," in *Computer Vision, 2005. ICCV 2005. Tenth IEEE International Conference on*, 2005, pp. 212-219 Vol. 1.

[33] B. Jahne and H. Haubecker, *Computer vision and applications*: Academic Press, 2000.

[34] J. Shin, S. Kim, S. Kang, S.-W. Lee, J. Paik, B. Abidi, and M. Abidi, "Optical flow-based real-time object tracking using non-prior training active feature model," *Real-Time Imaging,* vol. 11, pp. 204-218, 2005.

[35] P. M. H. I.A.Karaulova, A.D.Marshall, "A Hierarchical Model of Dynamics for Tracking People with a Single Video Camera," in *British Machine Vision Conf.*, 2000, pp. 262-352.

[36] L. Shin-Ping, C. Yuan-Hsin, and W. Bing-Fei, "A Real-Time Multiple-Vehicle Detection and Tracking System with Prior Occlusion Detection and Resolution, and Prior Queue Detection and Resolution," in *Pattern Recognition, 2006. ICPR 2006. 18th International Conference on*, 2006, pp. 828-831.

[37] A. Yilmaz, L. Xin, and M. Shah, "Contour-based object tracking with occlusion handling in video acquired using mobile cameras," *Pattern Analysis and Machine Intelligence, IEEE Transactions on,* vol. 26, pp. 1531-1536, 2004.

[38] J. K. Aggarwal and Q. Cai, "Human motion analysis: a review," in *Nonrigid and Articulated Motion Workshop, 1997. Proceedings., IEEE*, 1997, pp. 90-102.

[39] A. Ali and J. K. Aggarwal, "Segmentation and recognition of continuous human activity," in *Detection and Recognition of Events in Video, 2001. Proceedings. IEEE Workshop on*, 2001, pp. 28-35.

[40] T. Huang, D. Koller, J. Malik, G. Ogasawara, B. Rao, S. Russell, and J. Weber, "Automatic symbolic traffic scene analysis using belief networks," in *Proceedings of the Twelfth National Conference on Artificial Intelligence* Seattle, 1994.

[41] D. Koller, K. Danilidis, and H.-H. Nagel, "Model-based object tracking in monocular image sequences of road traffic scenes " *Int. J. Comput. Vision* vol. 10, pp. 257-281 1993.

[42] L. Goncalves, E. D. Bernardo, E. Ursella, and P. Perona, "Monocular tracking of the human arm in 3D " in *Proceedings of the Fifth International Conference on Computer Vision* IEEE Computer Society, 1995.

[43] K. Rangarajan and M. Shah, "Establishing motion correspondence," in *Computer Vision and Pattern Recognition, 1991. Proceedings CVPR '91., IEEE Computer Society Conference on*, 1991, pp. 103-108.

[44] C. J. Veenman, M. J. T. Reinders, and E. Backer, "Resolving motion correspondence for densely moving points," *Pattern Analysis and Machine Intelligence, IEEE Transactions on,* vol. 23, pp. 54-72, 2001.

[45] J. Heikkila and O. Silven, "A real-time system for monitoring of cyclists and pedestrians," in *Visual Surveillance, 1999. Second IEEE Workshop on, (VS'99)*, 1999, pp. 74-81.

[46] M. S. Arulampalam, S. Maskell, N. Gordon, and T. Clapp, "A tutorial on particle filters for online nonlinear/non-Gaussian Bayesian tracking," *Signal Processing, IEEE Transactions on [see also Acoustics, Speech, and Signal Processing, IEEE Transactions on],* vol. 50, pp. 174-188, 2002.

[47] Particle filter, website http://en.wikipedia.org/wiki/Particle_filter

[48] I. J. Cox and S. L. Hingorani, "An efficient implementation of Reid's multiple hypothesis tracking algorithm and its evaluation for the purpose of visual tracking," *Pattern Analysis and Machine Intelligence, IEEE Transactions on,* vol. 18, pp. 138-150, 1996.

[49] G. YIHONG, "Integrated Object Detection and Tracking by Multiple Hypothesis Analysis," *NEC J Adv Technol,* vol. 2, pp. 13-18, 2005.

[50] D. Comaniciu, V. Ramesh, and P. Meer, "Kernel-based object tracking," *Pattern Analysis and Machine Intelligence, IEEE Transactions on,* vol. 25, pp. 564-577, 2003.

[51] L. M. Fuentes and S. A. Velastin, "People tracking in surveillance applications," *Image and Vision Computing,* vol. 24, pp. 1165-1171, 2006.

[52] Open Source Computer Vision Library Project, website http://www.intel.com/technology/computing/opencv/index.htm

[53] R. Fisher, J. Santos-Victor, and J. Crowley, "CAVIAR: Context Aware Vision using Image-based Active Recognition," 2002.