IMAGE PROCESSING RESOURCE ALLOCATION METHODS FOR MULTI-TARGET TRACKING OF DISMOUNTED TARGETS IN URBAN ENVIRONMENTS

THESIS

Jon P. Champion, Second Lieutenant, USAF

AFIT/GE/ENG/06-13

DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY

AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

AFIT/GE/ENG/06-13

IMAGE PROCESSING RESOURCE ALLOCATION METHODS FOR MULTI-TARGET TRACKING OF DISMOUNTED TARGETS IN URBAN ENVIRONMENTS

THESIS

Presented to the Faculty

Department of Electrical and Computer Engineering

Graduate School of Engineering and Management

Air Force Institute of Technology

Air University

Air Education and Training Command

In Partial Fulfillment of the Requirements for the

Degree of Master of Science in Electrical Engineering

Jon P. Champion, B.S. E.E.

Second Lieutenant, USAF

March 2006

AFIT/GE/ENG/06-13

IMAGE PROCESSING RESOURCE ALLOCATION METHODS FOR MULTI-TARGET TRACKING OF DISMOUNTED TARGETS IN URBAN ENVIRONMENTS

Jon P. Champion, B.S. E.E.

Second Lieutenant, USAF

Approved:

[illegible]	13 Mar 06
LtCol Juan Vasquez Ph.D. (Chairman)	Date
Peter S Maybeck	13 Mar 06
Dr. Peter Maybeck (Member)	Date
[illegible]	13 May 06
Dr. Guna Seetharaman (Member)	Date

Table of Contents

Page

List of Figures

Figure Page

List of Tables

AFIT/GE/ENG/06-13

Abstract

Dismounted targets can be tracked in urban environments with video sensors. Real-time systems are unable to process all of the imagery, demanding some method for prioritization of the processing resources. Furthermore, various segmentation algorithms exist within image processing, each algorithm possesses unique capabilities, and each algorithm has an associated computational cost. Additional complexity arises in the prioritization problem when targets become occluded (e.g., by a building) and when the targets are intermixed with other dismounted entities. This added complexity leads to the question "which portions of the scene warrant both low cost and high cost processing?" The approach presented in this thesis is to apply multi-target tracking techniques in conjunction with an integer programming optimization routine to determine optimal allocation of the video processing resources. This architecture results in feedback from the tracking routine to the image processing function which in turn enhances the ability of the tracker.

IMAGE PROCESSING RESOURCE ALLOCATION METHODS FOR MULTI-TARGET TRACKING OF DISMOUNTED TARGETS IN URBAN ENVIRONMENTS

I. Introduction

This thesis is concerned with the challenges posed in attempting to use video imagery to track multiple dismounted targets through an urban environment. Sponsored by Air Force Research Laboratory's Sensors Directorate, the interest is in generating results that promote the mission of the Air Force. When simply considering multiple target tracking, global track data must be maintained and used to gather available information regarding the target. Properly handling this data allows target information to be maintained over long periods of time. Improperly handling this data may cause target information to interfere with each other, confusing the tracker and any other systems that rely on such information. The question of how to handle multiple target scenarios has been explored extensively in other research (1, 4) and is a mature subject.

Image processing is another mature subject that is constantly evolving due to the benefits of advances in computer speed and memory capabilities (9, 10, 11). Image processing can be very computationally intensive. In some applications, as advances in computer hardware and software are made, a similar increase in the demands on the computer systems occurs. Since many tracking algorithms are able to use attribute data that may be gained from a video sensor, image processing systems may be asked to perform more operations on an image to acquire more detailed or more complete data. The high demands of image processing often stretch the resources of a computer attempting to gather data in real-time. If a computer is unable to perform all desired operations on all of the data, it becomes desirable to schedule or allocate these limited resources. This leads into the next topic.

Resource allocation problems are a topic within the field of Operations Research (5, 12). The study of optimal resource allocation is a driving force behind much research, allowing systems to obtain the most value for an amount of effort spent. Several linear programming methods have been developed and include methods to formulate the problem based on a wide variety of physical parameters. Physical or technological limitations may be included in the problem formulation and even entire classes of problems may be developed based on these constraints. For example, binary integer programming is a form of linear programming that makes decisions strictly on the solutions having a value of 0 or 1. This formulation is used extensively in this thesis.

The combination of the issues discussed above leads to a desire to achieve several goals. We must be able to track targets moving through urban environments, and the tracker must be capable of operating adequately despite the challenging conditions posed by that urban environment. The image processing algorithm must be capable of obtaining measurements whenever such information is available to the camera. However, it should exploit only those algorithms that are useful and in ways that provide the most amount of information for the least amount of image processing time. Image processing algorithms should be selected that provide information that can be useful for classification of the targets moving through a scene. A suitable mix of methods that provide attribute data on a target is needed. A combination of attributes can provide information that will increase confidence in the classification of a target.

1.1 Problem Statement

One set of the multiple target scenarios is the tracking of targets through an urban environment. Urban environments are characterized by several situations that inhibit direct observation of a target. Using video data, it is readily possible to track a target through a simple open environment. While infrared video systems may be part of an implemented system, this thesis will be concerned strictly with visible video data. Urban environments contain areas where it is impossible to obtain visual data from a target, starving a tracking algorithm of data. In any case, the desired target may be hidden from view. It will be the goal of this thesis to present a method that maximizes the ability of a tracking algorithm to gain target attribute data in an urban environment while reducing the amount of computational load the image processing algorithms put on a computer system. During an occlusion event, the goal will be to maintain the track of a dismount target and reacquire it when it emerges from occlusion.

1.2 Chapter 2

This chapter will introduce the theoretical concepts used through the rest of the thesis. Image processing methods are used extensively for the work in this thesis, as are target tracking methods. The mathematical methods that allow these operations to be performed will be shown. Chapter 2 will also contain the information describing the Integer Program which is the Operations Research concept used to provide the maximum benefit-per-cost solution to the problem of knowing where to apply image processing techniques.

1.3 Chapter 3

This chapter will detail the methods employed and how the theory was used to generate a new method for implementing video-based target tracking. More advanced image processing methods will be presented in this chapter. Also, this chapter will explore how the various parts of the overall system are combined to work with each other.

1.4 Chapter 4

Chapter 4 will examine the experimentation method used and the results generated using a variety of conditions. This chapter will detail the results from the computer experiments to determine how effectively the method described in Chapter 3 actually worked. This chapter will contain the results of a short Monte Carlo run using the algorithm to show how it performs under noisy conditions.

1.5 Chapter 5

The final chapter of the thesis will summarize the results of the experimentation. It will detail how the results coincided with the expected and desired results. It will also list topics for further investigation.

II. Background Theory

The purpose of this chapter is to relate the applicable theory and certain technical information that will be utilized or exploited through the rest of this thesis. This thesis primarily utilizes the fields of Target Tracking, Image Processing, and Operations Research. The applicable theory within each of these fields will be discussed thoroughly in this chapter, to include any illustrations to orient the reader to the pertinent concepts.

2.1 Target Tracking

Target tracking is typically accomplished by incorporating measurement data in conjunction with an understanding of the physical dynamics of the target and the physical capabilities of the sensor. The following sections describe the method implemented to accomplish that goal, the linear Kalman filter.

2.1.1 Kalman Filter. The linear Kalman filter is the optimal estimator for a linear system with known noise and dynamics characteristics (8). The Kalman filter operates by accepting incoming measurements from a sensor, and using those measurements and known statistical and physical characteristics of a system to predict the best estimate of the system state.

A primary quantity of interest is the target state vector denoted as $\mathbf{x}(t)$ with dimension n. A state-space representation of the behavior of the target is given by the following continuous-time linear system model:

$$\dot{\mathbf{x}} = \mathbf{F}(t)\mathbf{x}(t) + \mathbf{B}(t)\mathbf{u}(t) + \mathbf{G}(t)\mathbf{w}(t) \tag{2.1}$$

where the dynamics matrix, $\mathbf{F}(t)$, represents the physical dynamics of the target. The system input control matrix, $\mathbf{B}(t)$, represents how the various input controls given by the control vector, $\mathbf{u}(t)$, are physically manifested to the target state. The matrix $\mathbf{G}(t)$ represents the system dynamics noise distribution matrix. It relates the driving noise denoted by the vector $\mathbf{w}(t)$ to the target state.

The following discrete-time or sampled-data linear measurement model is used to represent sensor information provided to the tracking algorithm, which for the purpose of this thesis is provided via video data:

$$\mathbf{z}(t_i) = \mathbf{H}(t_i)\mathbf{x}(t_i) + \mathbf{v}(t_i) \tag{2.2}$$

where the vector $\mathbf{z}(t_i)$ represents the measurements as they are presented from the sensors. The matrix $\mathbf{H}(t_i)$ indicates the relationship between the target's state, $\mathbf{x}(t_i)$, and the readings made by the sensor. The vector $\mathbf{v}(t_i)$ represents additive sensor measurement noise. Two noise matrices, $\mathbf{Q}$ and $\mathbf{R}$, are related to $\mathbf{w}(t)$ and $\mathbf{v}(t_i)$, respectively, and are defined as:

$$\begin{aligned} E\{\mathbf{w}(t)\mathbf{w}(t+\tau)^T\} &= \mathbf{Q}\,\delta(\tau) \\ E\{\mathbf{v}(t_i)\mathbf{v}(t_j)^T\} &= \mathbf{R}\,\delta_{ij} \end{aligned}$$

where $\delta(\tau)$ is a Dirac delta function and δ_{ij} is a Kronecker delta function. Also, note that both noises are assumed to be zero mean, such that:

$$\begin{aligned} E\{\mathbf{w}(t)\} &= \mathbf{0} \\ E\{\mathbf{v}(t_i)\} &= \mathbf{0} \end{aligned}$$

Using these definitions, at any time (t_{i-1}), the conditional mean and conditional covariance of the target state are defined as:

$$\begin{aligned} \widehat{\mathbf{x}}(t_{i-1}^+) &\triangleq E\{\mathbf{x}(t_{i-1})|\mathbf{Z}(t_{i-1}) = \mathbf{Z}_{i-1}\} \\ \mathbf{P}(t_{i-1}^+) &\triangleq E\{[\mathbf{x}(t_{i-1}) - \widehat{\mathbf{x}}(t_{i-1}^+)][\mathbf{x}(t_{i-1}) - \widehat{\mathbf{x}}(t_{i-1}^+)]^T|\mathbf{Z}(t_{i-1}) = \mathbf{Z}_{i-1}\} \end{aligned} \tag{2.3}$$

where $\mathbf{Z}_{i-1}$ represents the measurement history through time t_{i-1}. Given the desire to implement the filter in a computer, it is often convenient to employ the equivalent discrete-time form of the system given by:

$$\mathbf{x}(t_i) = \mathbf{\Phi}(t_i, t_{i-1})\mathbf{x}(t_{i-1}) + \mathbf{B}_d(t_{i-1})\mathbf{u}(t_{i-1}) + \mathbf{w}_d(t_{i-1}) \tag{2.4}$$

$$\mathbf{z}(t_i) = \mathbf{H}(t_i)\mathbf{x}(t_i) + \mathbf{v}(t_i) \tag{2.5}$$

where $\mathbf{w}_d$ represents the discrete-time zero-mean, white dynamics driving noise. The subscript d attached to the matrix variables indicate that these are discrete-time functions. The state transition matrix, $\mathbf{\Phi}$**,** is defined as the solution to:

$$d\left[\mathbf{\Phi}(t, t_o)\right]/dt = \mathbf{F}(t)\mathbf{\Phi}(t, t_o) \tag{2.6}$$

$$\mathbf{\Phi}(t_o, t_o) = \mathbf{I} \tag{2.7}$$

This thesis will operate under the assumption that the noise processes $\mathbf{w}_d(\cdot,\cdot)$ and $\mathbf{v}(\cdot,\cdot)$ are uncorrelated. The equivalent discrete-time forms of $\mathbf{B}$ and $\mathbf{Q}$ given are by:

$$\mathbf{B}_d(t_i) = \int_{t_{i-1}}^{t_i} \mathbf{\Phi}(t_i, \tau)\mathbf{B}(\tau)d\tau \tag{2.8}$$

$$\mathbf{Q}_d(t_i) = \int_{t_{i-1}}^{t_i} \mathbf{\Phi}(t_i, \tau)\mathbf{G}(\tau)\mathbf{Q}(\tau)\mathbf{G}^T(\tau)\mathbf{\Phi}^T(t_i, \tau)d\tau$$

With the discrete-time definitions available, the state mean and covariance may be propagated to a point before a measurement is received using the equations:

$$\widehat{\mathbf{x}}(t_i^-) = \mathbf{\Phi}(t_i, t_{i-1})\widehat{\mathbf{x}}(t_{i-1}^+) + \mathbf{B}_d(t_{i-1})\mathbf{u}(t_{i-1}) \tag{2.9}$$

$$\mathbf{P}(t_i^-) = \mathbf{\Phi}(t_i, t_{i-1})\mathbf{P}(t_{i-1}^+)\mathbf{\Phi}^T(t_i, t_{i-1}) + \mathbf{Q}_d(t_{i-1}) \tag{2.10}$$

In Equations (2.9) and (2.10), the superscripts $-$ and $+$ are are used to distinguish the values of the statistics just prior to, and just after, the incorporation of measurement information at a given time epoch, respectively. Following the propagations listed above, the state estimate and its associated covariance must be updated with the new measurement.

The Kalman gain, $\mathbf{K}(t_i)$, is computed via the relationship:

$$\mathbf{K}(t_i) = \mathbf{P}(t_i^-)\mathbf{H}(t_i)\left[\mathbf{H}(t_i)\mathbf{P}(t_i^-)\mathbf{H}^T(t_i) + \mathbf{R}(t_i)\right]^{-1} \tag{2.11}$$

Given this gain, the state mean and covariance may be updated with the new measurement information using:

$$\widehat{\mathbf{x}}(t_i^+) = \widehat{\mathbf{x}}(t_i^-) + \mathbf{K}(t_i)\left[\mathbf{z}_i - \mathbf{H}(t_i)\widehat{\mathbf{x}}(t_i^-)\right] \tag{2.12}$$

$$\mathbf{P}(t_i^+) = \mathbf{P}(t_i^-) - \mathbf{K}(t_i)\mathbf{H}(t_i)\mathbf{P}(t_i^-) \tag{2.13}$$

The following assumptions are required for the *linear* Kalman filter to be the optimal estimator in a statistical sense. The physical model of the system must be a linear model, driven by white Gaussian noise. While no physically realized system is truly linear, the Kalman filter will still perform adequately as an estimator provided the system is operated in a region wherein it is nearly linear. Otherwise, nonlinear filtering or approximations of nonlinear filters may be required (8). The assumption of a system driven by white Gaussian noise is also physically imperfect. However, in situations in which the noise is nearly white, a Kalman filter may still be used to provide adequate estimates of the target state. There are two general cases where this sort of situation may exist. In one case, the noise entering the system may appear to be white within the bandwidth of the system. In this case, the assumption is valid. Otherwise the noise may appear to be non-white. For the non-white noise case, linear shaping filters (8) that output non-white noise themselves but are driven by white Gaussian noise may be used to augment the system. In such a case, the assumption of a system driven by white Gaussian noise is valid.

2.1.2 Track Initiation. As a system receives measurements, there are two general possibilities to consider. The measurements may represent a target or they may be erroneous detections (4). A basic image processing technique is called change detection. This is the process of making comparisons between two or more frames of image data to determine where areas of change between the images have occurred. Change detection methods will be discussed more thoroughly in Section 3.5.1. In order to qualify as a track in the context of change detections via image processing, a measurement must have sufficient mass, that is, it must occupy enough pixels to be considered a dismount target. When a change area that has an appropriate mass is encountered, it is considered a candidate target. When available, image processing resources are assigned to the target to determine its attributes. When a track has sufficient attribute measurements representative of a target we wish to track, the track is initialized and classified in accordance with Section 4.1.5. Though this method is not based on a statistical analysis, for the cases studied in this thesis, it will be sufficient to initiate tracks. A multitude of track initiation methods are documented in the open literature (1, 4), but were not implemented due to time constraints on the research project.

2.1.3 Global Track Maintenance. The objective of proper global track maintenance is to ensure that, at each epoch, measurements that are being gathered are applied only to the target that generated the information. This is accomplished through proper association of tracks to measurements. Several methods have been devised to handle this aspect of tracking (1, 4). In general, the methods are categorized as hard or soft assignments. Typically, hard assignments are methods that make decisions that are firm, that is, are not intended to be reversed. This results in simpler computation, and is good for simple tracking scenarios with single or well separated targets. Soft assignment methods usually defer decisions for some number of epochs, while collecting more information and producing many hypotheses. These methods involve using probabilistic weighting to determine the track assignments and may be computationally intensive. For this thesis, a hard assignment method was employed to reduce computation time required to complete the track association. At each epoch, the measurement that is closest to the current track position estimate is used to update the measurement. Additional measurements, if any, are used to start new potential tracks. This method is referred to as nearest neighbor data association (4).

2.1.4 Data Collection and Utilization. Image attribute data defines the values of the critical components of the subject of interest. Target tracking concepts are employed to determine how best to exploit the acquired data. Certainly, acquired information will incorporate some degree of error as a result of discretization and measurement inaccuracy. With this knowledge, stochastic estimation via Kalman filtering can be applied to determine the new target state given the incoming data.

It is important to distinguish between kinematic and attribute data as well as kinematic and attribute states. Kinematic data is the information generated from change detection measurements (see Section 3.5.1). Target kinematic data includes the target's centroid as a measure of its current position. A Kalman filter produces the kinematic state by incorporating a series of measurements. In the case of attribute data such as color, texture, or mass that are fixed but contain uncertainties, one can employ set theoretical techniques such as Dempster-Shafer (6) or a simplified technique known as gamma weighting, discussed next.

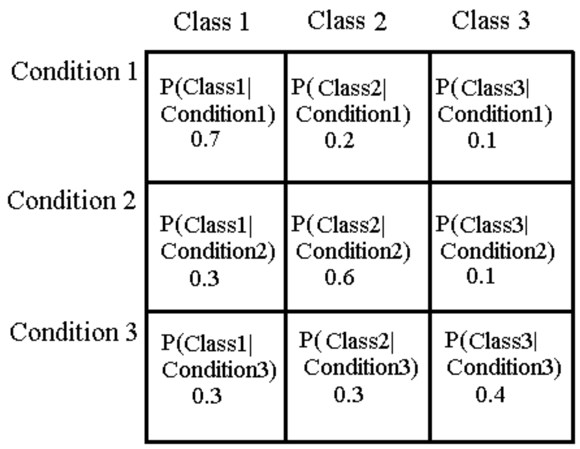
2.1.4.1 Gamma Weighting. Once measurements have been associated with a given track, it is important to incorporate that data with the current state of the track. Newly arriving data must be weighted properly when combining it with prior knowledge of a track. This process is done automatically in the Kalman filter. However, the Kalman filter requires detailed knowledge of the underlying statistical nature of both the target state and the measurements. The sensors do not have well-understood likelihood functions to describe the characteristics of the measurements. More specifically, the attribute measurements provided via image processing do not have definite uncertainty characteristics. This warrants a simplified sub-optimal approach. One sub-optimal approach is to use gamma-weighting. The method incorporates measurements as follows:

$$\widehat{\mathbf{x}}(t_i^+) = (1-\gamma)\widehat{\mathbf{x}}(t_{i-1}^+) + \gamma \mathbf{z}_i \tag{2.14}$$

where γ is an empirically determined scalar value. Gamma-weighting has the disadvantage that it does not utilize a covariance matrix, or many other Kalman filter concepts. However, when the Kalman filter cannot be properly defined due to unknown system or measurement noise characteristics, this may provide an adequate compromise. It should also be noted that the attributes are not propagated through time when gamma-weighting is used, unlike with the Kalman filter. One can readily see that γ is similar in nature to the Kalman gain used in the Kalman filter update Equations (2.11) through (2.13). However, γ is not computed based on underlying densities that represent the system model; γ is simply an empirical weighting.

2.1.4.2 Class Confusion Matrix. Classification of a target requires that the knowledge of a target's state be combined with knowledge of the probability of confusing the target of one class with that of another class. A target belongs to a class based on the presence of sets of characteristics that are observed via image processing algorithms. For this thesis, a target is considered to belong to one of several classes. Figure 2.1 provides a simple example of a classification problem with three target classes. The relative probabilities of each class being the correct class given a decision is shown. In this example, the probability of Class 1 being the declared identification given that the output is actually in Class 1 is 0.7, indicating that the system correctly identifies Class 1 objects relatively well. The figure indicates the probability of identifying Class 3 objects correctly is 0.4, with a significant chance of the object being misidentified as Class 1 or Class 2. Every row in the confusion matrix should add to 1, to indicate that these are exhaustive and mutually exclusive probabilities. There may be any number of rows and columns as appropriate to the problem. For example, if there are many more conditions that may be used for classification, or many more possible classes, the confusion matrix would have the proper number of rows or columns to account for conditions or available classifications. The purpose of the class confusion matrix is to help make decisions regarding the identification of the observed targets.

2.1.5 Target Identification. For this thesis, the conditions that define the rows of the class confusion matrix will be defined as a sum of several votes. These votes will be described in detail in Section 4.1.5. Given a time history of votes, $V_k = \{v_k,\ v_{k-1}\ \dots\ v_1\}$ used to determine the class, c_i, of a target, a Bayesian inference is used to compute the probability of an identification, ID, as $P_{ID} = p(c_i|V_k)$, for each track as follows (4):



	Class 1	Class 2	Class 3
Condition 1	P(Class1| Condition1) 0.7	P(Class2| Condition1) 0.2	P(Class3| Condition1) 0.1
Condition 2	P(Class1| Condition2) 0.3	P(Class2| Condition2) 0.6	P(Class3| Condition2) 0.1
Condition 3	P(Class1| Condition3) 0.3	P(Class2| Condition3) 0.3	P(Class3| Condition3) 0.4

Figure 2.1 Confusion Matrix Example

$$\begin{aligned} p(c_i|V_k) &= \frac{p\left(V_k|c_i\right)p(c_i)}{\sum\limits_j p\left(V_k|c_j\right)p(c_j)} \\ &= \frac{p\left(v_k|c_i\right)p(c_i|V_{k-1})}{\sum\limits_j p\left(v_k|c_j\right)p(c_j|V_{k-1})} \end{aligned} \tag{2.15}$$

where the discrete transitional density, $p\left(V_k|c_i\right)$, is based on attribute data using a simple binary voting method. The binary votes represent the set intersection of the various target features such as its characteristic color, texture, or mass. Note that Equation (2.15) assumes independence of the votes in the sense that:

$$p\left(V_k|c\right) = p\left(v_k, v_{k-1}, ..., v_1|c\right) = \prod_{i=1}^{k} p(v_i|c) \tag{2.16}$$

The declaration of target ID is made using maximum a posteriori (MAP) inference (1) at a given sample time via the logic decision:

$$ID = \arg\left\{\max_i p(c_i|V_k)\right\} \tag{2.17}$$

This hard decision is acceptable for many applications. It answers the question of "What class of target is it?" This classification can be used to drive various parts of the tracking algorithm. This is especially useful for situations in which system dynamics and noise attributes vary with target type. It may also drive optimization objective function coefficients or technological coefficients, which are discussed next.

2.2 Linear Programming

Linear programming is a mathematical method wherein a function's maximum value is obtained while a set of restrictions on a variable or set of variables are maintained. Linear Programming is also frequently used as a tool for the solution of optimization problems (12). Since the development of the first solution algorithm, the simplex algorithm, numerous other methods have been developed to allow for computer solutions to these problems. The following sections define three critical aspects of a problem's formulation—the Decision Variables, the Objective Function, and the Constraints. In addition, some mathematical characteristics of a linear programming problem will be explored, as well as two important subsets of linear programming problems.

2.2.1 Decision Variable. Decision variables are used in linear programming to define the solution space associated with the decisions to be made. The form of the decision variable helps to determine what type of problem is being solved. In general, linear programming decision variables can have any value. Within the context of integer programming, decision variables can only have integer or positive integer values. Binary integer programs, the type that will be used for optimization later in this work, have decision variables whose values may only be 0 or 1.

2.2.2 Objective Function. The objective function provides a mapping of the solution space to a numerical cost or benefit, which is ultimately minimized or maximized, respectively. Objective functions are defined by the characteristics of the problem being solved and may be linear or non-linear with respect to the decision variables. For linear programming problems, the objective function is strictly a linear combination of the decision variables. Maintaining this constraint on the problem formulation enables the use of many tools as well as a degree of simplicity within the solution procedure. For most linear programs, each decision variable makes one contribution to the value of the objective function. As a result, the objective function to be maximized or minimized may be written as:

$$J = \sum_i f_i d_i \tag{2.18}$$

where the values d_i represent the decision variables and each f_i represents a multiplicative objective function coefficient for each decision variable. The variable J is the cost or benefit to be minimized or maximized. Each objective function coefficient represents the contribution of its associated decision variable to the cost or benefit of the objective function.

2.2.3 Constraints. Constraints represent physical restrictions placed upon a problem. Ideally, with the above problems representation, if one were solving for a maximization or minimization, the decision variables would be made arbitrarily large or small. The constraints necessitate the optimization problem, as physical or practical limitations prevent the selection of decision variables of arbitrary size. It is necessary, due to the engineering or economic limitations on a problem, for certain limitations on the decision variables to be enforced within the linear program. Problem constraints account for these limitations and express them in terms of the decision variables. In this way, there will be a limit on the maximum or minimum value of any objective function. Constraints may appear as equalities, inequalities, or strict inequalities. An example constraint is:

$$\sum_i c_i d_i < N \tag{2.19}$$

The values c_i are often called technological coefficients, as they represent limitations due to the technology available.

2.2.4 Feasible Region. For linear programming problems with solutions, there is a set of points within which all of the linear programming constraints are met. This region is called the feasible region. A problem is considered feasible if and only if such a region exists. For problems having two decision variables, the feasible region can be depicted as a plane on which the constraint functions make cuts. With each cut through the plane, the feasible region becomes reduced, unless the constraint is superfluous. In such a case, the feasible region, and likewise the solution to the problem, are not affected by the constraint. Figure 2.2 shows an example of a linear programming problem. The constraints are shown as the lines that cut through the solution space. Two lines indicate that the solution must be greater than 0 for each variable x and y. These would likely be physical constraints based on variables not having negative quantities. Two other constraints ($y = 8 - 3x$ and $y = 4 - x$) indicate technological coefficients which can be written in the form of Equation (2.19).

2.2.5 Extreme Points. The feasible region is often an infinite set of points. This presents a challenge to finding the solution. However, it can be shown that if a linear program contains an optimal solution, then the optimal solution lies on one of the extreme points (12). An extreme point can be understood as the intersection of n inequality constraints in an n-dimensional space where none of the other inequality constraints are violated. While it is still possible for multiple or alternative solutions to a linear program will exist outside of the extreme points, if the problem has an optimal solution, at least one of the extreme points will be such a solution. Figure 2.2 illustrates a convex feasible region having several extreme points. These extreme points are illustrated in the figure as large dots at the end of the line segments that define the feasible region. Given any objective function expressed as:

$$J = f_1 d_1 + f_2 d_2 \tag{2.20}$$

the maximum value of J will exist at one of the extreme points. It is possible to contrive a problem for which the maximum J will exist along one of the constraint lines. For a two-dimensional problem, the extreme points may number as few as three points. The importance of this concept is a reduced search space for the problem solution with no loss in the chance of finding an optimal solution. The benefits of this technique are particularly important in higher-dimensional problems.

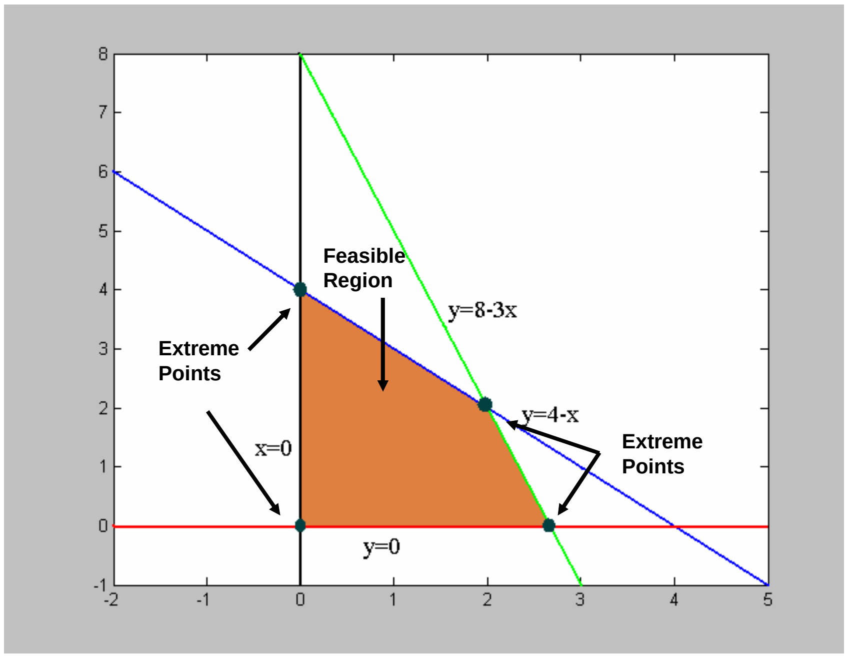


Figure 2.2 2-Dimensional Linear Program Example

2.2.6 Integer Programming. Integer Programming is a subset of linear programming in which the decision variables are constrained to having integer values (12). This is useful in situations in which the decision variables represent physical items which must remain integers. For example, a shipping company cannot task half of a truck to travel to one location, and the other half of the truck to another location simultaneously. Integer programming solutions cannot always be solved efficiently, so it is often employed as a tool for offline resource allocation. Binary integer programming represents another subset of problems within integer programming. Binary Integer Programming problems are such that the decision variables are constrained to two values, a 0 or a 1.

2.3 Image Processing

The techniques of image processing used within Chapters 3 and 4 are largely combinations of classical image processing methods (9, 11). The discussion here will cover aspects of morphological processing and color-space operations. Image processing using a computer is performed by representing the image as an array of numbers with length and width corresponding to the size of the image in pixels. A three-dimensional array may be

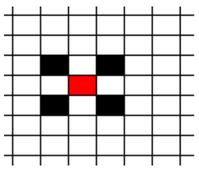


Figure 2.3 Dilation Structuring Element

used to represent more detailed attributes of a pixel. For instance, many color images are represented as two-dimensional rectangular arrays stacked three deep.

2.3.1 Binary Image. A binary image is represented by an array with values that are either 0 for black or 1 for white (9). The size of the array corresponds to the black and white image it represents. The utility of this form is that most basic image processing techniques are defined in terms of their effect on a binary image. Logical operations and set theory operations can be performed on binary images. Binary images may sometimes be referred to as a "mask." This indicates that the binary image is intended to represent an area that should or should not be considered for further operations. The image "masks" the area into or out of consideration.

2.3.2 Dilation and Erosion. Dilation and Erosion are a morphological image processing operations that "grow" and "reduce" a binary image (11). The type of growth or reduction is based on the shape present in a binary array called a structuring element. Every structuring element must have at least one element with a value of 1, and one “origin” element, which need not have a value of 1. Illustrations of dilation and erosion can be found in the accompanying figures. Figure 2.3 illustrates a structuring element example. The origin pixel is highlighted red for illustrative purposes. Figure 2.4 shows the seed image upon which the dilation operation will be performed and the pixel locations with a black square are assigned a value of 1. Figure 2.5 shows the result of the dilation function in which the structuring element is replicated at each seed location (those with a value of 1). The result is the "growth" of desired structures within the image.

Figure 2.6 provides an example of a structuring element to be used for the erosion operation, with the base pixel highlighted red. Figure 2.7 indicates the binary image that will be eroded. For illustrative purposes, this image has one area matching the pattern of

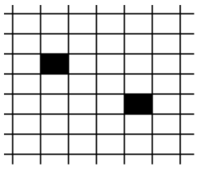


Figure 2.4 Dilation Seed Image

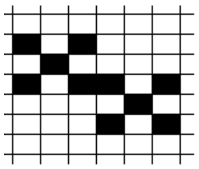


Figure 2.5 Dilation Result

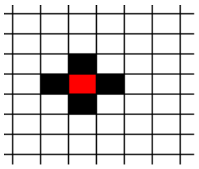


Figure 2.6 Erosion Structuring Element

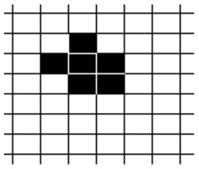


Figure 2.7 Seed Image for Erosion

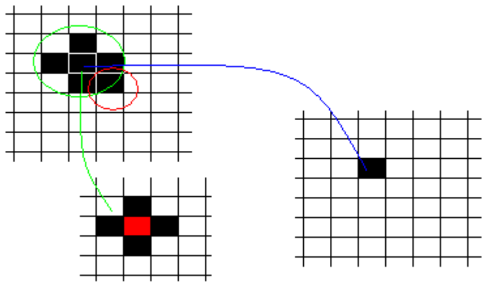


Figure 2.8 Erosion Process & Result

the erosion structuring element. In Figure 2.8, it can be seen that the array section that matched the structuring element is selected by the green circle. The center pixel has black neighbors that are identical to the arrangement of the structuring element. This center pixel will be the only pixel to be preserved after the erosion operation. The result of this is the "reduction" of the image except where the desired structures are present.

2.3.3 Opening and Closing. Combinations of dilation and erosion can facilitate image processing by removing excess noise from an image or filling in incomplete sections of an image (7). In this way, objects can be smoothed considerably, and even major deformations in an image can be resolved. This does require that some prior knowledge of what is expected from an image be presented and may require some judicious choice of structuring elements. When erosion followed by dilation (opening) is performed, or dilation followed by erosion (closing) is performed, it is common, but not necessary, to use the same structuring elements. Opening is used to break thin connections and remove thin protrusions from an image. Closing is useful for bridging thin gaps that are smaller than the structuring element. The implications of this in conjunction with target tracking in a binary image are discussed in Chapter 3.

2.3.4 Color Spaces. Color spaces are various ways of mathematically representing the full spectrum of depth of color that is visible to the human eye in three dimensions (9). Just as there are various ways of representing spatial coordinates, there are various methods of representing color. Two common color coordinate systems are the Red-Green-Blue (RGB) and Hue-Saturation-Intensity (HSI) coordinate systems. Visually, the HSI color space is represented as the set of cones in Figure 2.9.

In the RGB color space, color intensity data is determined by the intensity of each of the three primary additive colors for a given pixel. This is convenient for operations involving pure color addition, such as video displays, but is cumbersome when dealing with the information gathered by the human eye and describing the color characteristics of an object. The HSI color space better represents the desired information, as it separates the three components that define an object (10). Hue represents the actual color present and is represented in the HSI color space as a full circle. It begins at 0 with red, and

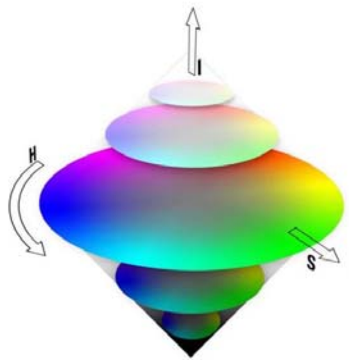


Figure 2.9 Representation of the HSI Color Space

progresses towards the yellow portion of the spectrum, then loops back around on itself with a value of 1, which is indistinguishable from the value of 0. This information precisely describes the color of a pixel. The saturation component describes the level of grayness of a pixel. Saturation values of 0 represent a pixel that is fully gray, and all grays of same intensity are indistinguishable regardless of their hue component. Saturation values of 1 represent a full color component, and are represented in Figure 2.9 along the outside of the circles. Intensity of a pixel represents the illumination placed on the pixel. A brightly illuminated pixel with a high intensity will appear white with progressively more color depending on the corresponding saturation component of the pixel. Low intensity pixels will appear nearly black, with intensity of 0 representing black, regardless of the values of the other two components. Transformations between the HSI and RGB color spaces are simple, invertible functions. The transformation from RGB to HSI is performed using the equations (9):

$$Hue = \cos^{-1}\left\{\frac{1}{2}\frac{[(R-G)+(R-B)]}{\sqrt{(R-G)^2+(R-B)(G-B)}}\right\} \tag{2.21}$$

$$Sat = 1 - \frac{3\min(R,G,B)}{(R+G+B)} \tag{2.22}$$

$$Intensity = \frac{1}{3}(R+G+B) \tag{2.23}$$

2.3.5 Texture. The intensity component of an image, though not necessarily useful for color description, does have the characteristic of being useful for describing an image's texture. This attribute can then be a descriptor of such an image. Image texture is a regional descriptor of image data and two forms of texture will be considered. One set deals with the intensity histogram data of a region. The other set deals with the Fourier spectrum of the array of image data (9).

When dealing with the histographic data of a color space, it is necessary to have sufficient color depth for the image. This allows the binning of the histogram to be based strictly on the unmodified data from the image and helps to make the texture data useful. Several statistical values can be extracted from the histographic information. These include the mean and standard deviation, which use the classical definitions of these two descriptors. Also included are a measure of smoothness and the skewness of the histogram. The numerical derivation for this information also uses the classical definitions of mean, standard deviation, or skewness. An image or region can be defined by the characteristics, which will provide a quantified measure of texture that can be used for further operations or comparison purposes.

Spatial measures are based on the Fourier spectrum, and have their use in distinguishing between periodic and non-periodic patterns. One can also use spectral data to quantify the differences between periodic patterns. In this way, subjects which exhibit periodic components in their intensity patterns will show distinguishing traits when their spectral data is quantified, and will likewise set themselves apart from irregular subjects in a way that regular histogram data may not otherwise define.

2.4 Summary

The three disciplines examined in this thesis are Target Tracking, Linear Programming and Image Processing. The theory used for this thesis is mature and does not require additional development for what it is used herein. The next chapter will show how the knowledge from these disciplines is combined to produce a system that achieves the goals mentioned in Chapter 1.

III. Concept Implementation

This chapter describes the implementation of the theoretical concepts discussed in Chapter 2. Some of the specific implementations of the theory used are included along with the ways in which methods are combined in order to provide the data that will be explored in Chapter 4.

3.1 System Overview

The goal of the system is to achieve optimal allocation of the image processing resources available to the computer. The cost for this optimal solution is the use of a complex algorithm that may itself consume significant resources. Therefore, a trade-off occurs between the resources required to perform the image processing, and those required to determine the optimal resource allocation solution. In order to compare the performance of an optimally performing system with a decidedly suboptimal system, two methods will be used to allocate the image processing resources. One will determine the optimal solution using a Binary Integer Program (BIP). The other will use simple logic decisions to determine the best image processing allocation, and is henceforth referred to as "Logic." This algorithm will be described in detail in Section 4.1.7.

Several components will work together to achieve a time-based feedback loop. In this loop, data acquired at an epoch will drive image processing resource allocation decisions at a future epoch. To implement an optimal resource allocation algorithm, the system is separated into several components that will operate together within any given iteration of the image processing loop. Each of the required conceptual tools needed to achieve the desired performance will be implemented at the appropriate point in the loop. Each of these tools is discussed in greater detail below.

3.1.1 General System Components. The components that are required for a complete processing loop are a kinematic tracking and data association algorithm, an image region valuation mechanism, an optimizer, a set of image processing algorithms and an attribute update and target classification function. When simple logic decisions are used for resource allocation, the program includes the kinematic tracking algorithm

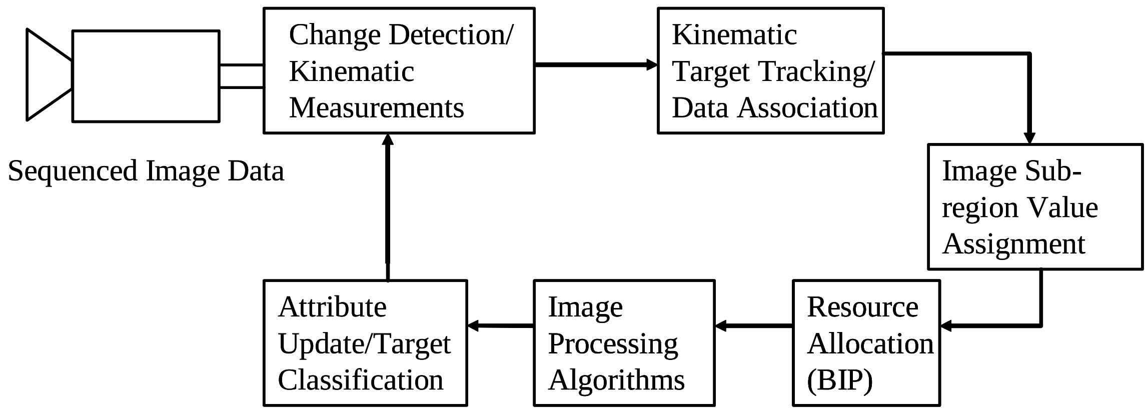

Change Detection/ Kinematic Measurements
Kinematic Target Tracking/ Data Association
Image Sub-region Value Assignment
Sequenced Image Data
Attribute Update/Target Classification
Image Processing Algorithms
Resource Allocation (BIP)

Figure 3.1 System with Binary Integer Program

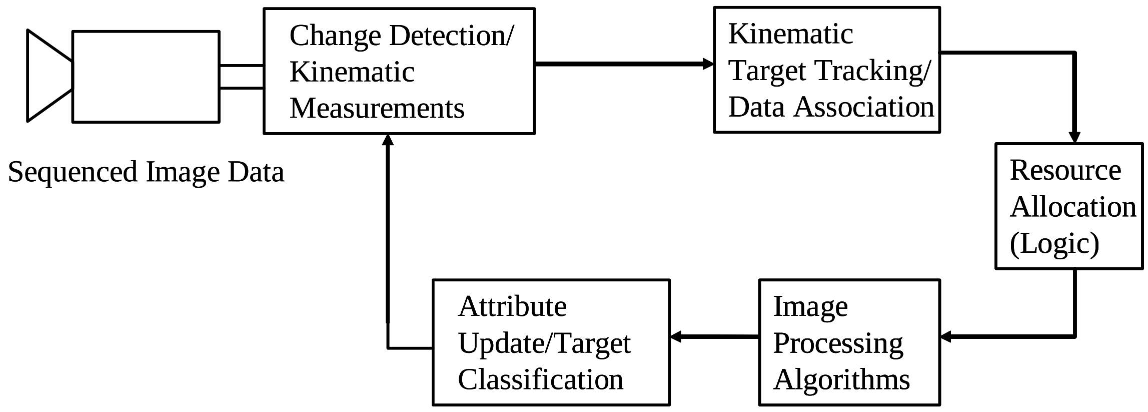

Change Detection/ Kinematic Measurements
Kinematic Target Tracking/ Data Association
Sequenced Image Data
Resource Allocation (Logic)
Attribute Update/Target Classification
Image Processing Algorithms

Figure 3.2 System with Logic Algorithm

and data association algorithm, image processing algorithms and attribute update and target classification functions, but does not have the image region valuation mechanism or an optimizer. These two components will be replaced with one block that makes basic decisions as to where to allocate the image processing resources. Figure 3.1 shows the overall layout of the system that uses the BIP. Figure 3.2 shows the system that uses the logic-based resource allocator.

The goal is to operate in such a manner that attribute and kinematic data are obtained from the image. This data is provided to the data association algorithm and tracking algorithm. For the BIP implementation, the information obtained from these two algorithms provides information about the field of view that is then provided to yet another algorithm that determines the relative value of performing image processing on various sections of the image at different resolutions. The BIP uses this value data to determine which areas of the image should be processed, at what resolution, and using which image-processing algorithms. The target attributes are updated leading to potential changes in target classification. Finally, target kinematic states are used to focus change detection at the next time epoch. The following sections will provide details for each of the blocks in Figures 3.1 and 3.2.

3.1.1.1 Change Detection/Kinematic Measurements. The first operation at an epoch is a change detection that is performed on the current epoch's image with respect to a background image. This change detection is restricted in scope to the commands given by the resource allocator at the previous epoch. The allocation is based on information provided by the kinematic state predictions, described in the next section. The kinematic measurements include the centroid of each region of change that appears in the image, and its mass. These measurements will provide the target position data that is used to update the kinematic tracks as described in the next section and Section 3.2.

3.1.1.2 Kinematic Target Tracking/Data Association. The target tracking algorithm consists of a Kalman filter to act as the discrete-time optimal estimator for the predicted state of any detected target given all previous knowledge of the target's kinematic behavior. The model used for the Kalman filter assumes that the target maintains a constant-velocity for its kinematic behavior over short time periods. Data association is the process of using the knowledge gained from previous measurements to determine which of the current measurements belong to which targets. Data association determines whether new measurements correspond to current tracks, whether they should be made into new tracks, or discarded altogether.

3.1.1.3 Sub-Region Value Assignment. One goal of this research is to determine which image processing algorithms should be applied to which sub-regions of an image. To determine objective function coefficients, each sub-region of an image should have a value associated with it to describe the benefit obtained from performing a segmentation algorithm on a sub-region. This section of the algorithm may require tuning as it relies on knowledge of the computational resources used by the equipment when assigning the proper values, as well as knowledge of the relative benefit of finding and properly identifying each target as one of the various target classes. When the value of each sub-region of the image has been determined, these values will drive the results of the binary integer program across the dimensions of position, resolution, and segmentation algorithm. Since the Logic-based resource allocator does not require sub-region value information, this will not be performed except when a BIP will determine resource allocation.

3.1.1.4 Resource Allocator. When attempting optimal resource allocation, the resource allocator that is used is the binary integer program. A BIP is formulated because the problem consists of many options that must either be used or neglected, that is, the options are binary. The advantages of using binary integer programming are a simpler problem formulation and a quicker solution using standard optimization techniques. It is possible to produce a "lead-in" optimization that would provide some parameters for the binary integer program to use for its constraint variables. Such a lead-in would be an integer program in which the decision variables would provide numbers of iterations of each segmentation algorithm that should be permitted, and possibly alter objective function coefficients. This demands that careful problem formulation be maintained in order to avoid creating infeasible problems during an iteration of the loop. This algorithm is compared against a resource allocator that uses simple, logic-based decisions to determine resource allocation as shown in Figure 3.2.

3.1.1.5 Image Processing Algorithms. The image processing algorithms are used to detect targets in the image and to generate information regarding these targets. The video source for this project is a color image generated within a computer simulation of an urban environment. The targets are large enough to have significant color components of their own, as well as some texture attributes. The algorithms chosen consist of a mass-determination algorithm, a color-space measurement algorithm, and a texture identification algorithm. Each algorithm was designed to provide data regarding only the region to which it is directed. Various other segmentation algorithms exist and could provide useful data. The three listed here simply provide a basis for the thesis study. As indicated in the system diagram, the change detection algorithm will be performed first. After the resource allocator has been run, the color and texture measurements will be performed.

3.1.1.6 Attribute Update/Target Classification. The target should be classified based on its attributes and how those attributes relate to a model of the desired target's attributes. The information obtained from this classification process is used for other purposes, including assigning value to some region around a target's physical space for the purposes of determining the value of image processing on such a region. The method used to determine the target class is the class confusion matrix method. This method provides a convenient simulation tool that preserves previously gathered information in a Bayesian sense while still permitting new information to have a significant impact on the outcome. The result is "binned" in the sense that the classification can be declared to belong to one of several possible options.

3.2 Target Tracking

The target tracking portion of the process involves several components for proper operation. At each epoch, kinematic measurements are made that must be identified as either a new target, part of an existing target, or some non-target measurement. This data association is for the target kinematics and an existing target predicted location will only be updated with its nearest neighbor measurement. If attribute measurements are assigned by the resource allocation algorithm, the target will gather a record of the attributes that have been measured and associated with it. A target can be associated with multiple attribute measurements. These measured attributes for a target will be sequentially updated with every attribute measurement that is associated with the target as described in Section 2.1.4.1.

3.2.1 Kinematic Tracking. Kinematic target tracking is based on the measurements received at a single epoch. Initial measurements are based on objects that differ from the background, and any such differences are be considered measurements. For this reason, an image will first experience a change detection based on the target kinematics before further attribute information will be gathered. Any measurements that are gathered are filtered using opening and closing techniques, as described in Section 2.3.3 in order to remove noise and other small erroneous measurements. The centroid of every measurement, $[z_x \; z_y]^T$ is tested for the possibility that it is the measurement for a target that already exists. Data association is conducted via a nearest neighbor method, where a measurement's distance from a target's predicted position, $[p_x \; p_y]^T$ is defined in terms of the position states:

$$\rho^z_{k,m} = \sqrt{(p_{x_k} - z_{x_m})^2 + (p_{y_k} - z_{y_m})^2} \tag{3.1}$$

for each target k and measurement m. The superscript z indicates this distance is related to measurement data. This is the association that is used to assign attribute measurements to tracks. This implies that a track which is updated with attribute data must first be associated with a kinematic update. Change detection measurements that do not have targets associated with them are initialized as new targets. The location of all targets is stored for the next stage of the target tracking process.

3.2.2 Bayesian Filtering. Since there is a possibility of measurement error at every epoch, a Kalman filter is used to update every target's kinematic state estimate. A discrete-time form of the Kalman filter is used to determine the new updated state and covariance estimate for each target based on the prior state and covariance estimate and the new measurement. The target state is propagated one epoch into the future to determine where the target is expected to be found at the next epoch. This estimate is maintained so that change detection resource allocation for the future epoch can be made. In effect, this predicts where the next set of change detections are likely to occur.

3.2.3 Track Deletion. Measurement updates may not occur at every sample time due to occlusions or resources not being applied to a target. As such, during periods of prolonged occlusion, the target track errors will grow. The reappearance of the target on a boundary of the occlusion would provide a measurement update for the existing track. However, there is no guarantee that the target will reappear by the time the propagated track reaches a boundary of the occlusion. This leads the to need to manage track deletion. Tracks are monitored to determine their propagated position and the length of time since no measurements were associated with them. When target propagations take them well beyond the range of the image and no measurements have been associated with them for an empirically determined number of epochs, the target's information is deleted and the target is considered to have vacated the scene. Alternatively, one could delete tracks based on the covariance of the estimated target state. The idea is that once track error grows to an excessive value (e.g., 10σ), then the track would be considered invalid.

3.3 Region Value Assignment

It is important to divide the entire image into discrete areas that can be used individually to determine which corresponding part of the image should be processed. For this, the image will be divided into many sub-regions. Each sub-region will be indexed according to its overall position in the image. There will be several layers of sub-regions that overlap each other. Large and small sized sub-regions will have the same number of total pixels. This allows larger sub-regions to cover more area in the image at the cost of reduced resolution. A detailed discussion of these aspects is given in Section 4.1.3. The sub-regions provide a convenient means to formulate the optimization problem with discrete decision variables.

Region valuations begin after the predictions have been made for each target's location at the next measurement epoch. As such, it is possible to assign a value for performing a segmentation algorithm on a sub-region of the image in the next frame. The value of performing an operation on a target varies with how much of the information that operation can provide and what degree of confidence we have in that information. For this reason, as long as the location of a target is updated frequently, it may not be necessary to update every characteristic at every epoch. A region's value assignment is also based on the amount and what type of information has been acquired in the past on that particular target. It is logical to consider these factors, since the optimization routine is capable of determining not only what areas to segment, but also which algorithms to use.

In order to classify a target properly, sufficient information must be collected to determine its attributes. In a system in which the information provided by the various segmentation algorithms is independent, it will be necessary for every segmentation algorithm to run on each target to be able to classify a target fully. In cases in which the attributes are highly correlated through time, it would be reasonable to exploit these correlations in classifying a target. In order to reduce the effects of false readings made by anomalous measurements, each target should experience each segmentation algorithm as often as possible. This goal competes with the desire to determine all the measurable attributes of all the targets given the limitation on processing resources. Given the independent nature of the attribute measurements used in this study and the correlation of attributes through time, an overall goal is to obtain variety in the measurements versus repetition of a single type of measurement. The sub-region value assignment algorithm will strive to meet this goal by granting more importance to target-segmentation combinations that have not yet been performed. It will also give more importance to target-segmentation combinations that have not been performed as often as others. One of the major goals is to obtain as much information as possible about targets that are of certain classifications. Therefore, sub-regions that are expected to contain such targets should have a higher value. Finally, it is necessary to determine how the information about these targets, their locations, and current classifications should be combined to create the coefficients that will be used to solve the binary integer program. The following discussion presents the method used to obtain the objectives just mentioned. The expression used to define the values of each sub-region and segmentation combination is:

$$f'_{i,j,k} = dist_{i,j,k} \cdot \kappa_k \cdot m_{j,k} \cdot \beta_j \tag{3.2}$$

where the coefficients will be described presently. Note that this function is a construct intended to convey information that could be important to describing the objective function coefficients.

The Euclidean distance between a track's current position and the center of each sub-region is used to begin calculating the value function for each i^{th} sub-region and for each k^{th} target. This distance is scaled in each axis by the target's velocity to account for the anticipated displacement at the next sample time. Since the value is intended to show the relative value of the target appearing at a given location, a Gaussian distance may be used to approximate that probability. In order to generate a Gaussian-like weighting of the value, the error distance, $\rho^s_{i,j,k}$, is used as the argument of the error function given by:

$$dist_{i,j,k} = \frac{2}{\sqrt{\pi}} e^{(\rho^s_{i,j,k})^2} \int_{\rho^s_{i,j,k}}^{\infty} e^{-t^2} dt \tag{3.3}$$

Where

$$\rho^s_{i,j,k} = \sqrt{(p_{x_k} - s_{x_i})^2 + (p_{y_k} - s_{y_i})^2} \tag{3.4}$$

with s_{x_i} and s_{y_i} representing the center of sub-region i. The superscript s indicates that this distance is from the center of a sub-region to the target. For the three segmentation case used in this thesis, the index $j = 1$ represents the change detection, and $j = 2, 3$ is used for the high-cost algorithms, color and texture measurements. It must also be computed for each sub-region, i, and will provide an array of values $dist_{i,j,k}$ that is computed through Equation (3.3).

Next, the target ID is used to provide a scale factor. Given the finite set of target ID's, a scale factor, κ_k, is applied for target k in order to put added emphasis on a particular type of classification. This will help to ensure that resources are not spent on targets that are of no consequence. For example, based on a set of possible ID's, the value κ_k may be defined as:

$$\kappa_k = \begin{cases} 2, & ID = 1 \\ 5, & ID = 2 \\ 3, & ID = 3 \end{cases} \tag{3.5}$$

The values for κ_k are arbitrary and represent an ad hoc tuning parameter within the value assignment constraint. Next, given a desire to obtain various types of attribute data via the j segmentation algorithms, an additional scaling of $m_{j,k}$ is given by the ratio between the number of times any segmentation j has been performed on a given track k to the number of times any other operation has been performed to this same track. This weight can be illustrated as:

$$m_{j,k} = \begin{cases} 1, & j = 1 \\ \frac{N_{2,k}}{N_{3,k}}, \quad N_{2,k} > N_{3,k}, & j = 2,3 \\ \frac{N_{3,k}}{N_{2,k}}, \quad N_{2,k} < N_{3,k}, & j = 2,3 \end{cases} \tag{3.6}$$

where $N_{j,k}$ is the number of times the j^{th} segmentation has been performed on the k^{th} target.

For the three-segmentation case used for this thesis, a relative weight, β, is now applied to discern the benefit of performing one type of processing over the other in terms of the benefit obtained in classification that results from this type of processing. For example,

$$\beta_j = \begin{cases} 1, & j = 1, & mass \\ 2, & j = 2, & color \\ 3, & j = 3, & texture \end{cases}$$

The values β_j are arbitrary and represent an additional ad hoc tuning parameter within the value assignment construct. Finally, since the goal is to determine a value for each sub-region and each segmentation algorithm, we will sum the values associated with each target using the expression:

$$f_{i,j} = \sum_k f'_{i,j,k} \tag{3.7}$$

to arrive at an objective function coefficient for each decision variable in the BIP.

3.4 Optimization Routine

The optimization algorithm uses a binary integer program to generate the optimal resource allocation solution for the next epoch. The decision variables for a binary integer program correspond to whether an action is performed or not. For a problem in which many combinations of actions can be taken, a binary integer program can have as many decision variables as there are legitimate combinations of all the possible actions. In systems where computational complexity is a concern, the BIP must be carefully formulated. A BIP can have 2^n possible solutions, where n is the number of decision variables. As a result, small increases in the size any of the factor under consideration can have tremendous implications on the computational complexity of a binary integer program. Fortunately, some solution techniques have built-in mechanisms to minimize this effect. In any case, every additional term should be considered for the contribution it makes to the overall benefit gained from performing this sort of optimization.

We formally define the binary integer program as follows:

$$\begin{aligned} \text{maximize} \quad & J = \sum_i \sum_j f_{i,j} d_{i,j} \\ \text{such that} \quad & A d_{i,j} \leq b \\ \text{and} \quad & A_{eq} d_{i,j} = b_{eq} \end{aligned} \tag{3.8}$$

where

$$\begin{aligned} d_{i,j} &= \text{binary decision variable for sub-region } i \\ & \quad\ \text{and segmentation method } j \\ f_{i,j} &= \text{value function at } i, j \\ A, A_{eq} &= \text{constraint matrices for limits on } d_{i,j} \\ b, b_{eq} &= \text{limit vector for the constraints} \end{aligned}$$

For the definitions of this program, it should be noted that a vector inequality is understood to mean all of the implied scalar inequalities are invoked simultaneously. Note that $d_{i,j}$ is a vector of length equal to the number of image sub-regions across all resolution levels (n_{sub}) multiplied by the number of segmentation methods (n_{seg}) considered. The value function follows the computation given in Equations (3.2) through (3.7).

The inequality constraints shown in Equation (3.8) are used to provide upper bounds on the number of decision variables with a value of 1 and thus limit the number of times that a sub-region and segmentation method can be selected. This is shown in Figure 3.3. The structure of the A-matrix is such that the first row in conjunction with the first element of the limit vector, b, will provide a limit on the number of times segmentation method #1 can be performed. A similar row is added for each segmentation method. Next, a row is

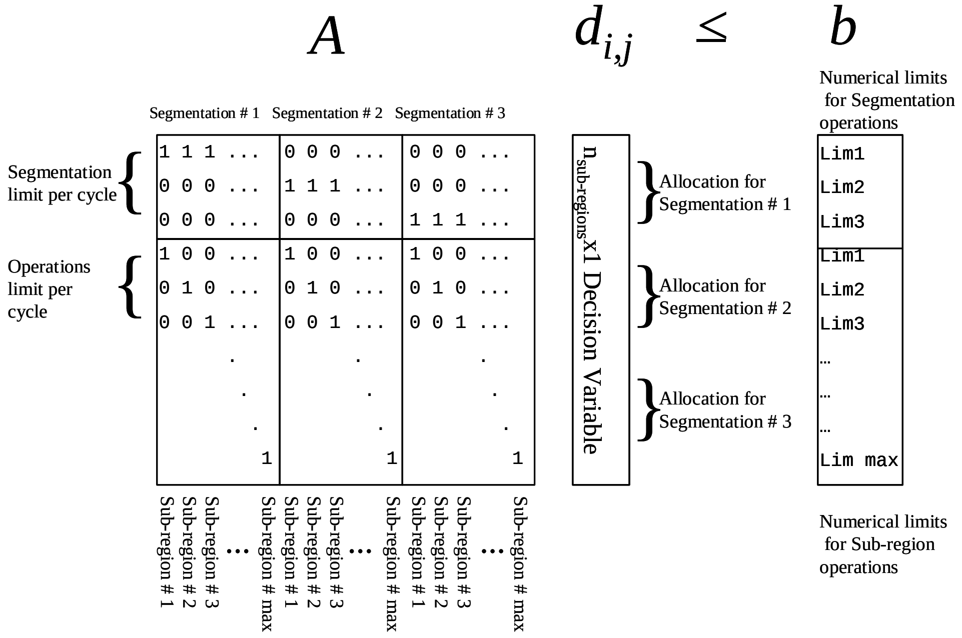

A
d_{i,j}
≤
b
Numerical limits for Segmentation operations
Segmentation # 1
Segmentation # 2
Segmentation # 3
Segmentation limit per cycle
Operations limit per cycle
n_sub-regions x1 Decision Variable
Allocation for Segmentation # 1
Allocation for Segmentation # 2
Allocation for Segmentation # 3
Lim1
Lim2
Lim3
Lim max
Sub-region # 1
Sub-region # 2
Sub-region # 3
Sub-region # max
Numerical limits for Sub-region operations

Figure 3.3 BIP Formulation, including forms of matrix A and vector b

added for each sub-region in conjunction with another element of the vector b to provide a limit on the number of times each sub-region can be operated on at each epoch. The dimensions of the A-matrix will be $(n_{sub} + n_{seg})\,\mathrm{x}\,(n_{sub} * n_{seg})$.

The second set of constraints is used to ensure that the optimization provides a greedy search, that is to say it will use all of the processing time available at each sample time. This is accomplished by setting b_{eq} equal to the maximum allowable cost which represents the total processing time available. Also, the elements of A_{eq}, are based on the time required to perform each of the possible segmentation tasks. The form of A_{eq} is given by:

$$\begin{bmatrix} c_1 & \overset{n_{sub}}{\cdots} & c_1 & c_2 & \overset{n_{sub}}{\cdots} & c_2 & c_{n_{seg}} & \overset{n_{sub}}{\cdots} & c_{n_{seg}} \end{bmatrix}$$

where $c_s =$ the cost to perform segmentation method s. The dimensions of the A_{eq}-matrix will be 1 x $(n_{sub} * n_{seg})$.

3.5 Image Processing Algorithms

Data provided by the video sensor provides useful information about the various targets. The purpose of the image processing algorithms is to extract this information and provide it to the system. The computational complexity of any image processing technique is often prohibitively high if it is performed on an entire image. If the area of an image can be reduced to some fraction of its original size, the computational load to extract the same information is likewise reduced. Furthermore, by extracting only the part of the image of concern via masking, it is possible to remove non-target information prior to performing a segmentation algorithm and improving classification. This masking is the first type of segmentation algorithm used in this research. Other segmentation algorithms involve extracting data regarding the color and texture characteristics of a target. Details of these algorithms will now be discussed.

3.5.1 Change Detection. In order to determine where targets are moving within a frame, two common methods may be employed(7). One method is to examine a set of three or more frames in a sequence. If the images have the same background scene, then pixels that are different between the three images will be objects that have moved. The second method is to use a known background image file to make comparisons with the current time epoch.

3.5.1.1 Three-Framed Change Detection. In order to identify the location of targets in a given frame, the two previous frames may be used in conjunction with the current frame. For this discussion, they will be called Frames A, B, and C. The difference between Frames A and B will provide areas where targets existed in the previous two epochs, but it will not be possible to distinguish which objects were present in which frame. By considering the difference between Frames B and C, it is possible to identify where different pixel regions exist between these frames. This difference can be compared with the difference between Frames A and B. Objects that are in the difference between Frames B and C, and are not in the difference between Frames A and B, are the current locations of moving objects. Assuming proper registration of the sequential frames, a change detection is applied between Frames A and B to produce Frame $A - B$. This

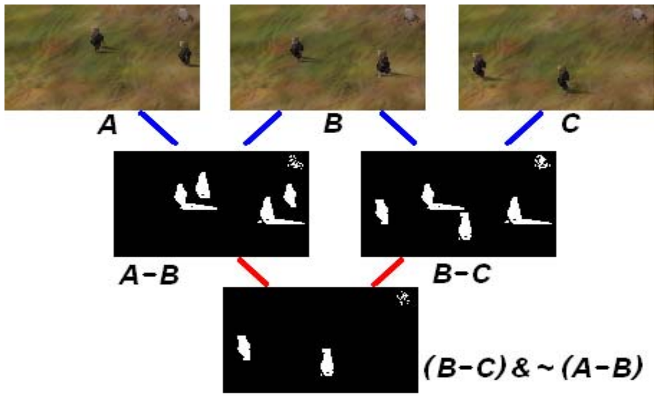


Figure 3.4 Example of Three-Framed Change Detection

difference is shown as the white areas and is called the change mask. The same process is applied to Frames B and C. The inverse of the change mask $A - B$ results in white areas (logical 1s) where there were no changes before. A change detection is observed when the inverse mask of $A - B$, denoted as $\sim(A - B)$ undergoes the AND operation with the mask of $B - C$. The output is the image at the bottom of the figure. This bottom image is the location of the moving targets in Frame C. The advantage of this method is that it does not require prior knowledge of the background image. A disadvantage is that this method requires that the targets be in continual motion.

3.5.1.2 Known Background Change Detection. A simple method of change detection is to examine the current image against a known background image. This method detects changes between the two images without the need for a sequence of images. Additionally, it may distinguish idle objects. However, this method does require that the scene be previously mapped and the background extracted in order to operate. This simplifies the process of identifying regions of change between images in order to identify moving objects. Known background change detection will be used for the results discussed in Chapter 4.

3.5.2 HSI Color Characteristics. If a color measurement is required for a sub-region of the image, a change detection mask is applied to this sub-region to extract pixels that pertain to a valid moving object. Each of these pixels is then converted from the RGB color space to the HSI color space. Histographic representations of the color space allow a probability density function (pdf) of the measured object to be evaluated regardless of the object's complexity (10). This pdf describes the amount of each color that is expected to occur in a target. The hue and saturation components, *hue* and *sat*, are represented as histographic data by discretizing the component values into equally spaced bins. Recall that both *hue* and *sat* take on values over the range $[0, 1]$ such that discretizing them over a finite number of bins, n_{bins}, would be represented as:

$$0 \leq hue < \frac{1}{n_{bins}}; \frac{1}{n_{bins}} \leq hue < \frac{2}{n_{bins}}; \ldots \frac{n_{bins}-1}{n_{bins}} \leq hue \leq 1 \tag{3.9}$$

$$0 \leq sat < \frac{1}{n_{bins}}; \frac{1}{n_{bins}} \leq sat < \frac{2}{n_{bins}}; \ldots \frac{n_{bins}-1}{n_{bins}} \leq sat \leq 1 \tag{3.10}$$

noting that each component may have a different number of bins.

3.5.3 Texture Characteristics.

An image's texture is measured using two different methods for this study. One method exploits statistical measures of the histographic data of the intensity part of the HSI color space. A second method uses spatial Fourier transform data to generate statistical measures.

3.5.3.1 Texture Histographic Measures.

As with hue and saturation, the intensity data can be discretized and binned to generate a histogram denoted as $p(z_i)$ and computed via:

$$0 \leq intensity < \frac{1}{n_{bins}}; \frac{1}{n_{bins}} \leq intensity < \frac{2}{n_{bins}}; \ldots \frac{n_{bins}-1}{n_{bins}} \leq intensity \leq 1 \tag{3.11}$$

However, this is not used specifically for this thesis, as the intensity space is already binned into 256 increments representing the 8-bit resolution provided in the conversion to the HSI color space. Regardless of whether the space is binned further, the following operations can be performed.

When a target has been completely defined in its intensity space, many attributes can be readily gained. Three of these attributes will be used in this study. These are the mean, $\mu_{texture}$, corresponding to average intensity; standard deviation, $\sigma_{texture}$, to measure average contrast; and the third moment, $\gamma_{texture}$, to measure the skewness of the histographic data. The combination of these three moments generates the attribute data that will be used for classification when statistical texture is used.

The mean, $\mu_{texture}$, is computed using:

$$\mu_{texture} = \sum_{i=0}^{L-1} z_i p(z_i) \tag{3.12}$$

where z_i represents the intensity at a point, $p(z_i)$ is the histogram of the intensity levels in the region, and L represents the number of possible intensity levels. For standard 24-bit color images, L will be 256. The standard deviation, $\sigma_{texture}$, is simply the square root of the second central moment of the intensity values, and the third central moment, $\gamma_{texture}$, is given by

$$\gamma_{texture} = \sum_{i=0}^{L-1} (z_i - \mu_{texture})^3 p(z_i) \tag{3.13}$$

3.5.3.2 Texture Spectral Measures. Another set of measures of texture are based on a spatial Fourier transform. This transform is capable of extracting frequency data for the image. This is useful for image patterns that contain frequency-based information such as periodic patterns. It does not provide significant useful information to distinguish between seemingly random patterns. Additionally, there is a danger that, if a target is partially occluded in a frame, the spatial information could be corrupted and provide false information.

As with histographic texture, spectral texture operates on the intensity component of the target being observed. Spectral measurements are a result of the 2-dimensional Fast Fourier Transform (FFT) of the image. The result of the FFT are intensity values, $S(r, \theta)$, in polar coordinates. For each direction, θ, $S(r, \theta)$ is changed to a 1-dimensional function, $S_\theta(r)$. The same is performed for each distance, to obtain a 1-dimensional function $S_r(\theta)$. The functions become a global descriptor through the summation of the functions (9):

$$S(r) = \sum_{\theta=0}^{\pi} S_\theta(r) \tag{3.14}$$

$$S(\theta) = \sum_{r=1}^{R_0} S_r(\theta) \tag{3.15}$$

where R_0 is the radius of a circle centered at the origin. Varying the pair of coordinates, (r, θ), allows consideration of a region of the image or the entire image. The typical statistical descriptors for these functions are the peak values, the mean and variance for each of the two functions, and the distance between the mean and the highest value for each function.

3.5.4 Filtering Corrupted Images. Change detection methods may be extended to images that have been corrupted(7). This requires some knowledge as to the type of corruption to be useful. These methods may use information regarding the magnitude of difference between images, opening and closing techniques, or a combination of the two. The purpose of this filtering is to ensure that change detections used to create a change mask are actual changes and not simply noise within the image.

3.5.4.1 Magnitude of Change Filtering. For the purposes of this thesis, the corruption method was to compress the digital image using a lossy compression mechanism. When extracted, most of an image's pixels will have been modified from the original image. In order to correct for this, a vector-difference method was used to determine "changed" pixels. Each pixel was considered a three-element array. The difference between a background image pixel and the corresponding current image pixel was calculated as a magnitude and compared against some cutoff. Pixels that exceeded this value were considered changed pixels and part of the change mask, otherwise the changes were considered to be noise and not part of the change mask.

3.5.4.2 Close-Open Filtering. Recall from Section 2.3.3, opening and closing are combinations of erosion and dilation. Opening involves performing an erosion operation followed by a dilation operation (7). Closing is the opposite; performing a dilation followed by an erosion. Small noise regions can be eliminated using opening. This includes speckling, or "salt-and-pepper" noise, that may be common depending on the video sensor. Opening can also be used to break narrow isthmuses within images, which may often be the result of objects being close together, and may smooth boundaries between an object and the background. Performing closing on an image helps to fill holes within a region caused by noise in the image, and is useful if the object contains small, disconnected regions.

3.6 Attribute Updating and Target Classification

The final block shown in Figures 3.1 and 3.2 show the attribute updates and target classification that will be described in this section.

3.6.1 Attribute Update.

At each epoch, the measurements obtained through the allocated image processing techniques may be associated with the targets that are present in the scene. When the measurement is the first measurement gathered for a target, then that attribute is initialized using the acquired information. Further measurements may be gathered at future epochs. To update the applicable attributes, the measurement is combined with the previously acquired attributes in accordance with Equation (2.14).

3.6.2 Target Classification.

The classification is based solely on target attribute data. Each attribute is compared to the attributes of a known desired target. For each attribute, a metric is determined that can be used to determine whether the target matches the desired characteristics. The metrics are described in detail in Section 4.1.5. The result of these comparisons is a set of votes that can be used in conjunction with a class confusion matrix. The result is calculated through the process discussed in Sections 2.1.5 and 4.1.5. The classification result may be used for a variety of purposes, but for this thesis, they are used to drive the value function generation in Equation (3.5). It is also used as a metric to determine the effectiveness of the resource allocation algorithms, as presented in Section 4.2.3.

3.7 Summary

This concludes the description of the computational mechanisms employed in this thesis. All of the devices described in this chapter will be employed in some way to produce the results found in Chapter 4. In the next chapter, the experiment will be described in which the BIP algorithm is compared to the Logic algorithm. The results of that experiment will be described in detail.

IV. Simulation Methods, Results and Analysis

This chapter will describe in detail the experimental methods used. This will include how the applicable theory is intended to drive the performance of the simulations. The results gathered from the experiments will be analyzed to determine the effectiveness of the algorithms proposed.

4.1 Simulation Methods

The experiment consists of a computer simulated urban region containing buildings, vehicles, and dismounts. The dismounts move through the scene in a pre-determined path, taking routes to simulate several of the difficult events that can occur with urban environment tracking. These events include:

1. A target occluded by a structure.
2. Two targets moving together in close proximity to each other.
3. Several targets within close proximity and with crossing paths.
4. A target that is idle for some period.

These are only a few of the possible events that can occur; however, they represent a general set of problems that involve dismounts.

4.1.1 Definitions. The following conventions will be used for the remainder of this thesis.

Definition 1 *Target: An object in the scene that generates measurements and forms a track.*

Definition 2 *Desired: The target we seek to track and classify properly.*

Definition 3 *Desired Classification: Proper declaration that the target exhibits attributes of the desired target.*

Definition 4 *Decoy: Any target which is not the desired target. Synonymous with non-desired.*

Definition 5 *Decoy Classification: Declaration that the target does not exhibit attributes of the desired target.*

Definition 6 *Unknown Classification: Declaration that the target's measured attributes do not warrant a desired or decoy classification. Note that "unknown" is only a classification and all targets are truly either the desired target or a decoy.*

Definition 7 *Scene: The image that contains the features of an urban environment through which objects may move.*

Definition 8 *Scenario: A sequence of images in which objects move for the purpose of the study of the program effectiveness.*

Definition 9 *Run: One Monte Carlo iteration of the program; several runs are performed on the same scenario.*

Definition 10 *Sub-region: A 40-by-30 pixel section of a scene or frame in a sequence of images.*

Definition 11 *Token: One unit of computing resources for image processing, roughly the computing power required to perform a change detection on a sub-region.*

4.1.2 Identifying Occlusion Region. In addition to gathering the sequential frames of the image, it is necessary to determine the background image and the occlusion mask. The occlusion mask is used for the valuation function explained in Section 4.1.6.1. The occlusion mask is designated by manually coloring parts of the image that contain occlusions. The color magenta is a popular choice for the "null color" as it does not appear by itself in any of the images gathered. Magenta pixels were mapped to a binary image mask that indicates where the occlusions occur in the image.

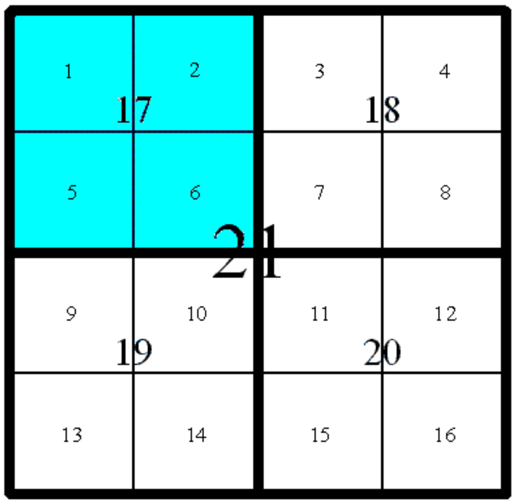


Figure 4.1 Sub-Region Example

4.1.3 Sub-Region Setup. A variety of methods were considered for deciding which parts and how much of an image should receive image processing resources. Input images were divided into equal-sized sub-regions, with each sub-region being given a number according to its position. In order to simulate lower resolution, square sets of four sub-regions at one level of resolution were combined into one lower level of resolution. In this way, sub-regions of lower resolution will overlap four of the higher-resolution sub-regions. Figure 4.1 shows an example arrangement of sub-regions for an image. In this figure, sub-regions 1 through 16 are at the highest available resolution and cover the smallest amount of image space. Sub-regions 17 through 20 are of an intermediate size, but can still be used for some image processing concepts. In this figure, sub-region 17 also encompasses information for sub-regions 1, 2, 5, and 6, but at lower resolution. The lowest level of resolution is shown as sub-region 21, which includes information for the entire image. This figure is used only for illustrative purposes. The actual sub-region map for the test images used in this thesis contain 341 sub-regions. Sub-regions 1 through 256 are the highest resolution level. The resolution then decreases by factors of two until one final low resolution sub-region that consists of the entire image is numbered.

4.1.4 Tracking Algorithm. The targets being simulated for this exercise are computer representations of dismounts as they move through an urban scene. The targets are colored images roughly the shape of a human when viewed from an elevated position. Data will be collected on the targets regarding the kinematic states and a set of attributes. The attributes will include mass, color, and texture characteristics.

4.1.4.1 Kinematic Tracking. Target models were designed to approximate human walking movement. An obvious characteristic is that the targets quickly accelerate to a constant speed and maintain this speed for the duration of their maneuver. Additionally, the targets may change direction quickly without slowing down or showing a change in the target's aspect. For these reasons, a constant velocity model is used to represent the targets. The constant velocity dynamics model will be used for the Kalman filter that is implemented as part of the tracking algorithm. The noise matrix, $\mathbf{G}_d(k)\mathbf{w}_d(k)$, is designed to accomplish good tracking of the constant velocity segments between epochs. Defining the kinematic state vector, $\mathbf{x}_{\text{kin}}(k) = [p_x \; v_x \; p_y \; v_y]^T$, the constant velocity model is given by (2):

$$\begin{aligned} \mathbf{x}_{\text{kin}}(k) &= \Phi(k, k-1)\mathbf{x}_{\text{kin}}(k-1) + \mathbf{G}_d(k-1)\mathbf{w}_d(k-1) & (4.1) \\ &= \begin{bmatrix} 1 & T & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & T \\ 0 & 0 & 0 & 1 \end{bmatrix} \mathbf{x}_{\text{kin}}(k-1) + \begin{bmatrix} \frac{T^2}{2} & 0 \\ T & 0 \\ 0 & \frac{T^2}{2} \\ 0 & T \end{bmatrix} \mathbf{w}_d(k-1) & (4.2) \end{aligned}$$

where T is the time between measurement points $(k-1)$ and k. The measurements consist of a two-element vector related to the state vector by the expression:

$$\begin{aligned} \mathbf{z}_{\text{kin}}(k) &= \mathbf{H}\mathbf{x}_{\text{kin}}(k) + \mathbf{v}(k) & (4.3) \\ \mathbf{z}_{\text{kin}}(k) &= \begin{bmatrix} z_x \\ z_y \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \mathbf{x}_{\text{kin}}(k) + \mathbf{v}(k) & (4.4) \end{aligned}$$

This implementation indicates that the two measurement axes are aligned with the "real" global axes. This is a rather trivial case, but it is sufficient to demonstrate the rest of the algorithm. For both of the above equations, $\mathbf{w}_d(k)$ and $\mathbf{v}(k)$ are two independent zero-mean white noise processes that satisfy Equation (**??**). Linear Kalman filtering will be used for the kinematic tracker using this simple target model. The Kalman filter will provide an estimate of the target's kinematic state at every epoch.

4.1.4.2 Attribute Tracking. At every epoch where change detections are made, the results are filtered using the methods described in Section 3.5.4 to generate a change detection mask. The MATLAB® command *bwlabel* is used to separate the regions of change into the masks that become the measurements used for the kinematic update. Afterwards, decisions are made by the resource allocation algorithm to determine the image processing resources that will be applied to each sub-region. When the image processing algorithms are run on a sub-region, attributes that are located in a target's change mask become the attribute measurements for that target. For this thesis, there are three modes of image processing: mass, color, and texture. Mass is the number of pixels a target's change mask occupies within the image. This number will vary for a target as it moves through the scene due to changes in pose and the extent of any target occlusion. However, there is a useful minimum and maximum mass corresponding to targets that may be considered dismounts. A target's mass attribute is initialized by simply taking the first measurement that is obtained. Another option for initializing the mass attribute would be to use an average mass value based on empirical data. The chosen method was to update the attribute using the gamma weighting discussed in Section 2.1.4.1:

$$\mathbf{x}_{mass}(k) = (1-\gamma)\mathbf{x}_{mass}(k-1) + \gamma\mathbf{z}_{mass}(k) \tag{4.5}$$

where $\mathbf{z}_{mass}(k)$ is the measured mass attribute at epoch k, and $\mathbf{x}_{mass}(k)$ is the updated mass attribute. An ad hoc value of $\gamma = 0.5$ was used to provide equal weight to past estimates and new measurements. Due to time constraints, a full empirical analysis was not conducted to determine an alternative value for γ.

All other attributes are updated in a similar manner. These attributes include the histographic data of the target's hue and saturation values to represent the color characteristics of a target. The texture values chosen to represent the target are the radial and angular spectral measures of texture given by Equations (3.14) and (3.15). These attribute characteristics are updated via gamma weighting ($\gamma = 0.5$) at each epoch in which measurements are actually taken. These sets of updates are used to alter the current attributes of the target for the purposes of classification which is described in the follow section.

4.1.5 Target Classification. Once a target's attribute characteristics are collected, it is possible to estimate the type of target being observed. Three classification states are considered: a desired state, a decoy state, and an unknown state. The unknown state is included as a means to help identify targets that may have several attributes related to the target, but for which it is not appropriate to make the "Desired" declaration. A model of the desired attributes was gathered before the simulation was run. At each epoch, the current state of each target's attributes is compared against the desired target. The mass is measured to determine whether it falls within two standard deviations of the desired target's mass. A measure of correlation (discussed momentarily) between a target's color and texture measurements and the desired target's color and texture components is used to determine if the target has the "Desired" attributes. Color and texture components are measured separately. Histographic hue measurements generate a correlation value when compared to the histographic hue data of the desired target. A similar comparison is done with histographic saturation data. If these components individually meet at least a prescribed correlation value, they are considered a match. For hue data, the correlation cutoff was set to 0.75 and the saturation cutoff was set to 0.5. These values were chosen by examining how well the desired target model correlates with itself across time and selecting a value that would obtain a high percentage of "yes" votes when the desired target is being measured. The same method is done with radial and angular spectral texture measurements. If *both* of those measurements exceed an empirical correlation value, then the texture component is considered a positive. The correlation values for these two characteristics were set to 0.9 for the $S(\theta)$ component and 0.98 for the $S(r)$ component. These values were chosen in similar fashion to the hue and saturation correlation cutoff values.

4.1.5.1 Voting. Comparisons between the measured target attributes and the desired target attributes provide four measures: mass, hue, saturation, and overall texture. The mass attribute is checked against the model of the desired target by determining if the measured mass falls within two standard deviations of the desired target's model. If it does, then the target receives a "yes" vote regarding how well the target matches the desired model. Comparison's of an object's hue, saturations, and texture measures, will be made by determining the linear correlation between the measured data and desired target model for each attribute. Let x be the measurement data, y be the desired target's data and n represent the number of bins as defined in Section 3.5.2, then the linear correlation coefficient, r, is defined as (3):

$$r = \frac{\sum_{i=1}^{n} x_i y_i - \frac{\left(\sum_{i=1}^{n} x_i\right)\left(\sum_{i=1}^{n} y_i\right)}{n}}{\sqrt{\left[\sum_{i=1}^{n} x_i^2 - \frac{\left(\sum_{i=1}^{n} x_i\right)^2}{n}\right]\left[\sum_{i=1}^{n} y_i^2 - \frac{\left(\sum_{i=1}^{n} y_i\right)^2}{n}\right]}} \tag{4.6}$$

Since r will have a value between -1 and 1, with high values indicating higher correlation between the target and the model, an empirical threshold can be used to determine or "vote" regarding how well the target matches the desired model. Each of these comparisons results in a binary vote to indicate whether or not the measure represents the desired target. This results in a total of 4 votes that are used to describe how well the measured target matches the desired target. The number of "yes" votes is used to determine the identification state of a class confusion matrix, as described in Section 2.1.4.2. Ideally, the class confusion matrix would represent the behavior experienced through a statistical analysis of the information. For this thesis, the matrix was generated arbitrarily and designed to achieve desired characteristics with the target classification declarations. The matrix used is:

$$\begin{array}{c c} Votes & \begin{array}{ccc} \text{Decoy} & \text{Unknown} & \text{Desired} \end{array} \\ \begin{array}{c} 0 \\ 1 \\ 2 \\ 3 \\ 4 \end{array} & \begin{bmatrix} 0.38 & 0.33 & 0.29 \\ 0.35 & 0.34 & 0.31 \\ 0.34 & 0.35 & 0.31 \\ 0.33 & 0.34 & 0.33 \\ 0.28 & 0.33 & 0.39 \end{bmatrix} \end{array} \tag{4.7}$$

This matrix contains three columns, representing the three classifications used in this simulation. From left to right, they represent the Decoy, Unknown, and Desired states. These classifications are represented notationally as c_i where $c = [Decoy\ Unknown\ Desired]$. The probability that a target is of class i is given by $p(c_i)$. The rows of the matrix represent the number of votes from zero to four, that the target received. Recalling Equation (2.17), and using the information from Equation (4.7), each target has three identification probability states that are determined by the values of $p(c_i)$. The values indicated by $p(V_k|c_i)$ in Equation (2.15) refer to the values in the row corresponding to the number of votes in Equation (4.7). To initialize the classification probabilities, the three scalar values $p(c_i)$ are set to 0.33 to indicate that there is no preference of one state over the other. Note that $p(c_i)$ is normalized at each time epoch via:

$$p(c_i) = \frac{p(c_i)}{\sum_i p(c_i)} \tag{4.8}$$

to ensure the probabilities sum to 1.

4.1.6 Optimization Routine. The optimization routine is the part of the program that examines the state of the targets, their classifications, and some other information regarding the targets to determine what the proper allocation of resources should be. For this thesis, two resource allocation methods are considered. The optimization routine is based on Binary Integer Programming, and will be referred heretofore as "BIP." A competing algorithm is discussed in Section 4.1.7.

4.1.6.1 Valuation Algorithm. There are 1023 total decision variables under consideration. Each image is of size 640-by-480 pixels, and is broken down into 256 sub-regions, each of size 40-by-30 pixels. These sizes were chosen to allow for sub-region sizes that could fully contain a simulated target, while allowing an integer number of sub-regions in both image dimensions. Furthermore, there are 5 resolution levels, with the lower resolution levels containing 64, 16, 4, and 1 sub-regions respectively. This produces 341 sub-regions for each available image processing method. Given that we consider three segmentation algorithms, this result is multiplied by 3 to produce 1023 total decision variables for the resource allocation methods to select. In order to determine the benefit of performing image processing on a certain region at a certain resolution level, a routine is used to compute the value of that sub-region using several factors. The result is an array containing an element for each combination of sub-region and image processing algorithm. This process is described in Section 3.3. Relating this to Equation (2.18), each combination of sub-region and segmentation corresponds to an i in the variable y_i.

Equations (3.2)-(3.7) describe the process that is used to develop the value function. The factors used for β were 2 for texture measurement and 3 for color measurement and 1 for change detection. These choices put an added emphasis on color measurements, which were found to be the most significant measurement made in the scenarios analyzed. For this thesis, the chosen values for κ were 2 for the "decoy" classification, 3 for the "desired" classification, and 5 for "unknown." This purpose of this is to divert resources to targets that have not been distinctly classified, and to apply more resources towards targets that appear to have attributes corresponding to the desired model versus a decoy. Note that the values for β and κ are ad hoc and arbitrarily chosen to achieve the desired performance, and that full analysis is needed to determine the values for best performance.

4.1.6.2 Constraints. In order to complete the development of the BIP, certain constraints must be placed on the problem. This will include the technological constraints described in Section 2.2.3 and system constraints as described in the last line of Equation (3.8). For this study, BIP is restricted to 34 change detections, 4 color measurements and 4 texture measurements. These constraints were arbitrarily chosen based on maximum number of targets expected to transit the scene. Furthermore, in order to avoid focusing all resources on one area, BIP is also restricted to using only two operations per sub-region. Finally, it is restricted to an arbitrary total of 70 tokens that it can apply. These ad hoc values were chosen to represent hypothetical limits on the amount of resources available per epoch.

4.1.7 Competing Algorithm. In order to compare the performance of the BIP, a competing algorithm was developed and tested to investigate how well it performs against the same data sets the BIP was given. The algorithm that competes with the BIP is a logic based algorithm, hereafter called "Logic." Logic will determine which sub-regions contain the propagated positions of the targets and assign change detections to those sub-regions as well as the eight adjacent sub-regions. Logic will also determine the sub-regions that contain estimated positions of all the targets and place one high-cost algorithm on that sub-region as well as the ones above, below, left and right. It will also assign the other high-cost operation to the sub-regions that are diagonal and connected to the sub-region that contains a target. This method was chosen to ensure that the sub-region containing the centroid of any target will receive image processing, along with any surrounding sub-regions, in the event that the target occupies an area across two sub-regions. The algorithm uses whatever resources are necessary to identify the targets in the image. It does not respect constraints on resources as the BIP does, nor does it have any sort of function to determine which sub-regions are most valuable for which processing algorithms. Clearly, the choices presented here are arbitrary, but they provide a simplistic method for resource allocation in regions close to the estimated position of the target.

4.1.8 Experimental Method. Two scenarios were generated for a 10-run Monte Carlo analysis. In each run, where each measurement point was generated, the location was perturbed by a white Gaussian noise of variance equal to 5 using the MATLAB® *randn* command. This arbitrary value was selected to provided statistical variation in the kinematic position data. The two resource allocation methods were compared, given the same trajectory data for a given Monte Carlo run. The statistics used to compare the performance of the two resource allocation methods are the classification accuracy that is generated from each run and the number of times each method uses an image processing algorithm.

4.2 Results

The scene through which the targets move is shown in Figure 4.2. This scene contains several buildings, two vehicles that do not move, and three different types of terrain that across which the targets will move. These features are chosen to represent the level of complexity of a typical urban scene. The buildings and targets will cast shadows that have

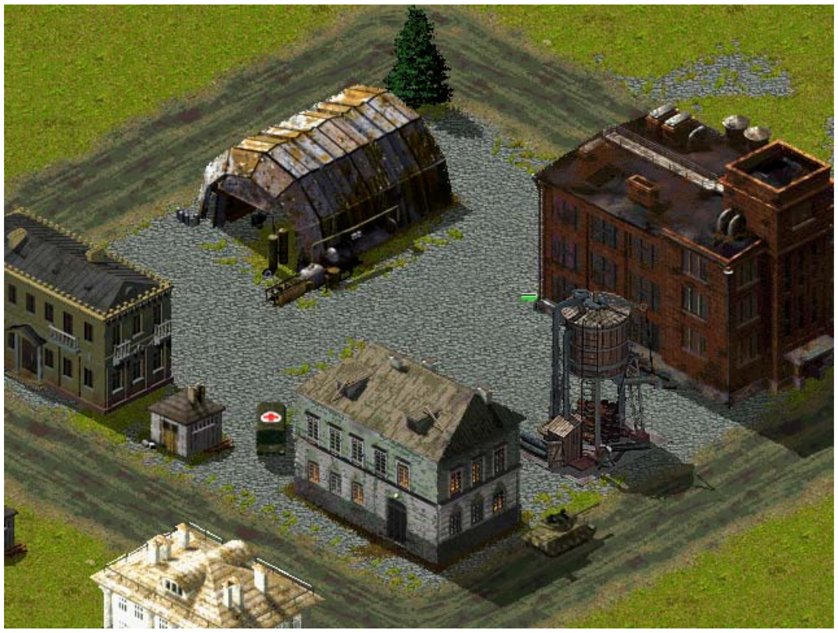


Figure 4.2 Background Scene for Simulations

different effects on the scene. The building's shadows may affect the targets, altering their appearance. The shadows of the targets may appear as part of the target, altering the apparent size of the target. This will be especially problematic if the target's shadow is included with the target's attribute measurements, as the data will not be representative of the target. These aspects will provide challenges to the classification process.

The paths taken by each target for Scenarios 1 and 2 are shown in Figures 4.3 through 4.8. The stars indicate the desired target's path. The circles and squares in these figures indicate the paths taken by the decoy targets. These figures will be used to compare the results generated by the two algorithms.

When evaluating the various simulation outputs, there are several important measures of performance. The algorithm must be able to classify the targets correctly, and the algorithm should use as few operations as possible. It should also provide a benefit in the incidence of detections versus false alarms. These measures were used to compare the performance of the BIP against the Logic algorithm.

Table 4.1 depicts the classification performance categories. The number of times each image processing algorithm was used by BIP and Logic is tabulated in Tables 4.2 and 4.3

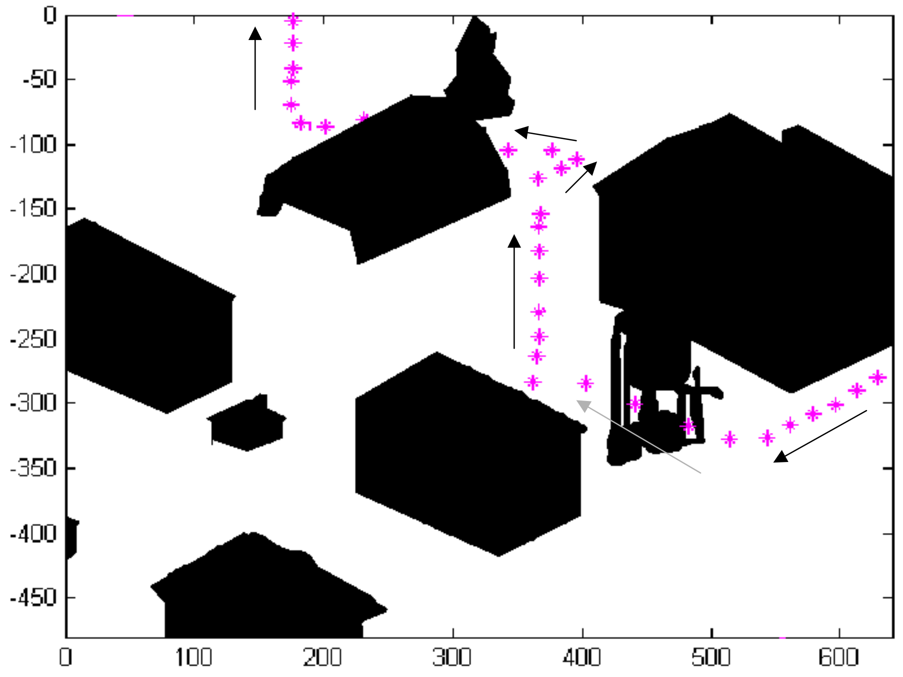


Figure 4.3 Scenario 1, Desired Target's Path

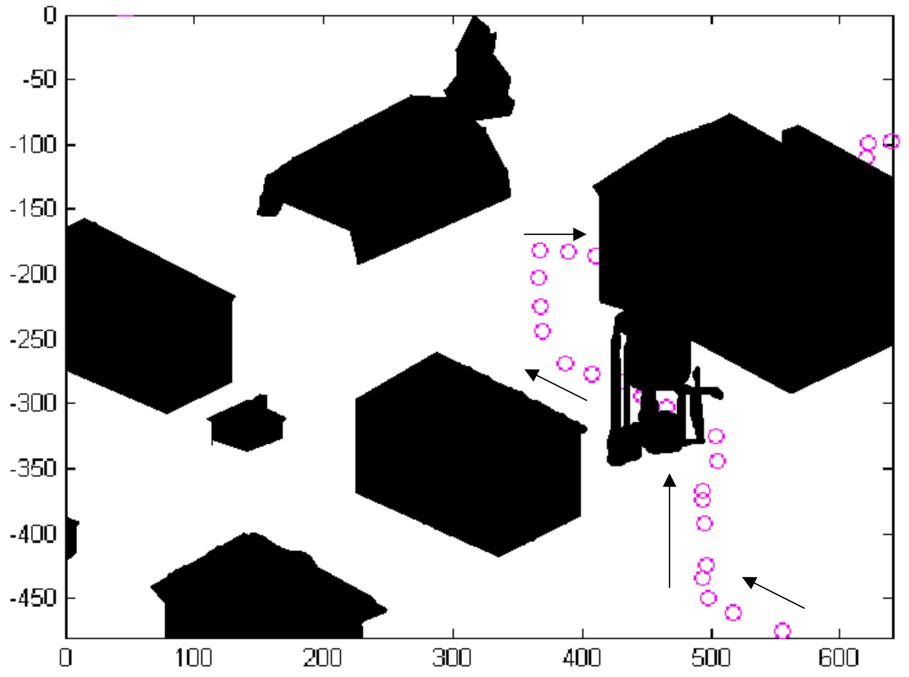


Figure 4.4 Scenario 1, Decoy #1's Path

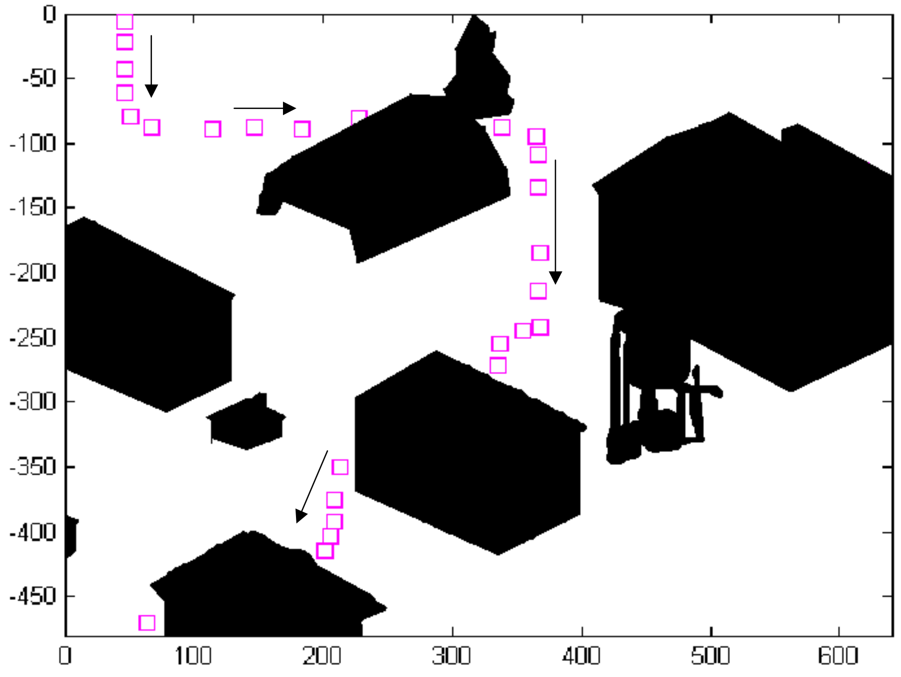


Figure 4.5 Scenario 1, Decoy #2's Path

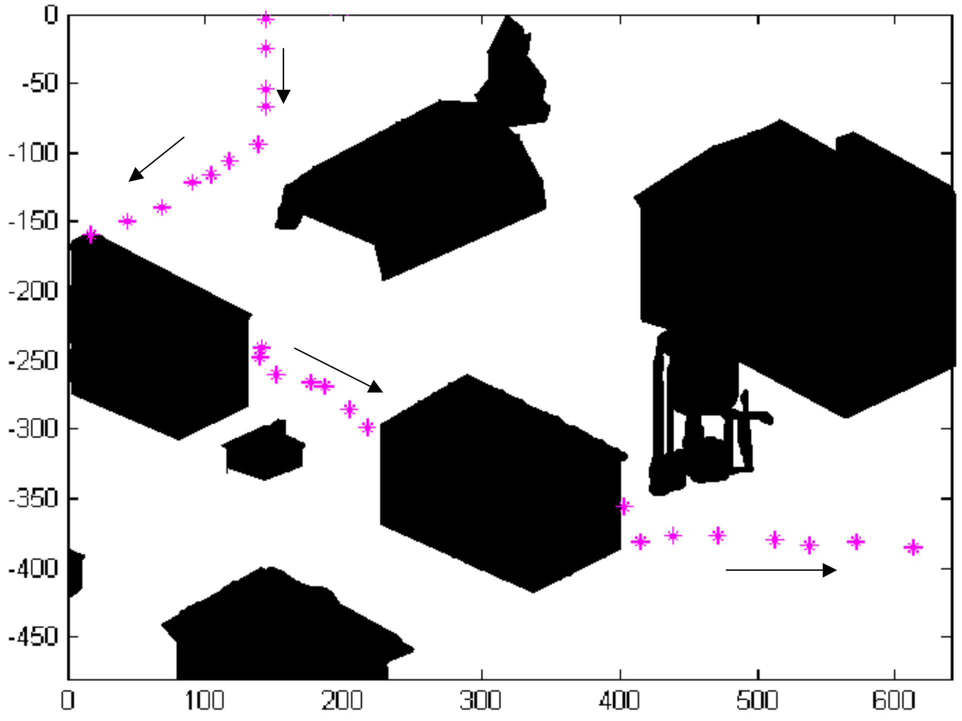


Figure 4.6 Scenario 2, Desired Target's Path

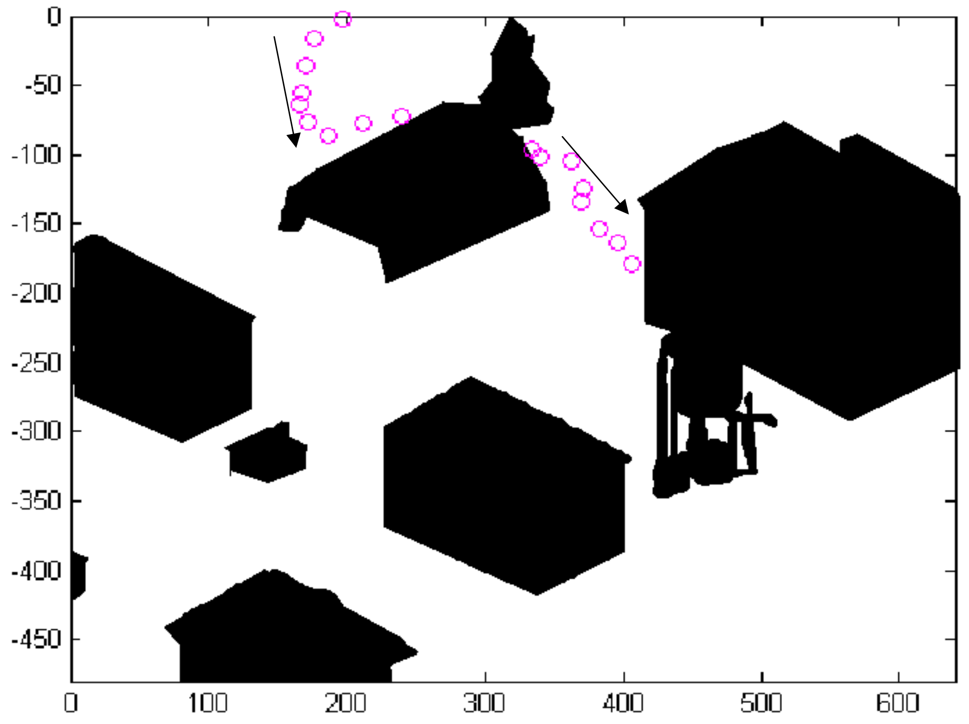


Figure 4.7 Scenario 2, Decoy #1's Path

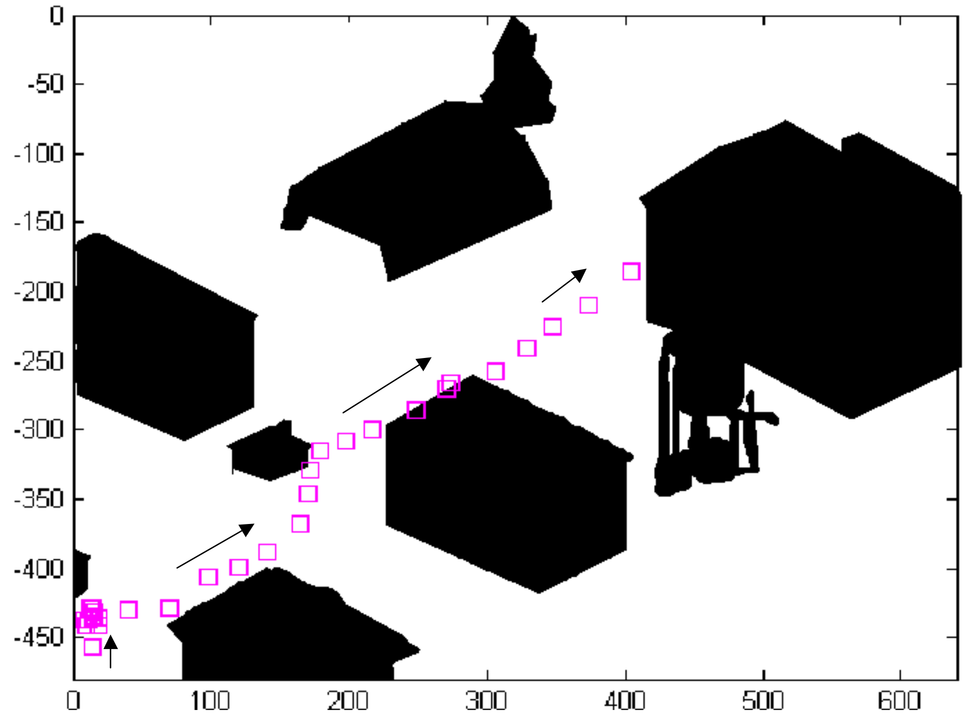


Figure 4.8 Scenario 2, Decoy #2's Path

Table 4.1 Classification Performance Categories

Target	Classification	Class Accuracy
Desired	Desired	True Positive
Desired	Decoy	False Negative
Decoy	Desired	False Positive
Decoy	Decoy	True Negative

Table 4.2 Resources Consumed by Allocation Methods for Scenario 1

BIP	Mass	Color	Texture	Logic	Mass	Color	Texture
	1808	189	184		1418	340	326
	1810	184	184		1435	345	338
	1812	204	185		1392	327	313
	1807	188	185		1404	329	323
	1808	184	184		1445	352	341
	1811	186	184		1419	335	332
	1806	186	184		1410	329	329
	1812	186	184		1418	335	331
	1811	184	184		1411	333	326
	1813	188	184		1445	348	345
Averages:	1809.8	187.9	337.3	Averages:	1419.7	337.3	330.4
Tokens:	1809.8	751.6	921	Tokens:	1419.7	1349.2	1652
	Total	Tokens:	3482.4		Total	Tokens:	4420.9
% Savings:	-27.4	44.2	44.2				
	Net	Savings:	21.2%				

for Scenarios 1 and 2, respectively. In order to compare classification performance, true and false positives and negatives are counted and tabulated. The relative results for each of the algorithms in Scenarios 1 and 2 are compiled in Table 4.6.

4.2.1 Algorithm Utilization. One of the goals of the resource allocator is to use fewer segmentation operations to classify a target than its competitor. Figures 4.9 through 4.14 show the number of iterations of each algorithm that are used at each epoch. The red dotted lines indicate Logic's usage, while the green dashed lines indicate BIP's usage. These figures show the average number of iterations over 10 Monte Carlo runs.

From these figures, it can be seen that the BIP restricts itself to using only the number of iterations that are given using the BIP problem formulation as described in Section

Table 4.3 Resources Consumed by Allocation Methods for Scenario 2

BIP	Mass	Color	Texture	Logic	Mass	Color	Texture
	2182	228	228		1612	342	342
	2170	230	228		1583	325	330
	2167	232	228		1594	330	336
	2180	228	228		1590	326	336
	2181	228	228		1588	331	329
	2178	230	228		1589	327	334
	2179	230	228		1658	363	367
	2183	232	228		1611	338	345
	2180	228	228		1590	327	335
	2183	230	228		1610	339	343
Averages:	2178.3	229.6	228	Averages:	1602.5	334.8	339.7
Tokens:	2178.3	918.4	1140	Tokens:	1602.5	1339.2	1698.5
	Total	Tokens:	3482.4		Total	Tokens:	4640.2
% Savings:	-35.9	21.4	32.9				
	Net	Savings:	8.7%				

4.1.6.2. Specifically, the values these figures indicate are those prescribed in Equation (3.8), with the BIP constrained to 34 change detections, 4 color measurements and 4 texture measurements. The trends observed in the Logic set indicate that the algorithm's usage increases in situations where there are more targets present, without regard for the amount of time required to perform those segmentation operations. The algorithm chooses to search more sub-regions in order to ensure that the target has been fully measured. The result is an algorithm that uses many more resources than BIP. By studying the amount of time required to perform each type of measurement, it was possible to determine the relative quantity of system resources required to perform each of the measurements. There are three segmentation algorithms used to generate the four measurements. The cost of performing a segmentation was compared to the time required to perform a change detection, as that is the fastest of the available segmentations. When using basic filtering on the images, it was determined that it takes about 5 times longer for a texture measurement and about 4 times longer for a color measurement than a mass measurement. A token is defined as the amount of time to perform a mass measurement. Therefore, for each segmentation, j, these relative token costs are called τ_j. For each segmentation that is performed v_j times per epoch, the total number of tokens, Ψ, required to perform an epoch's image processing

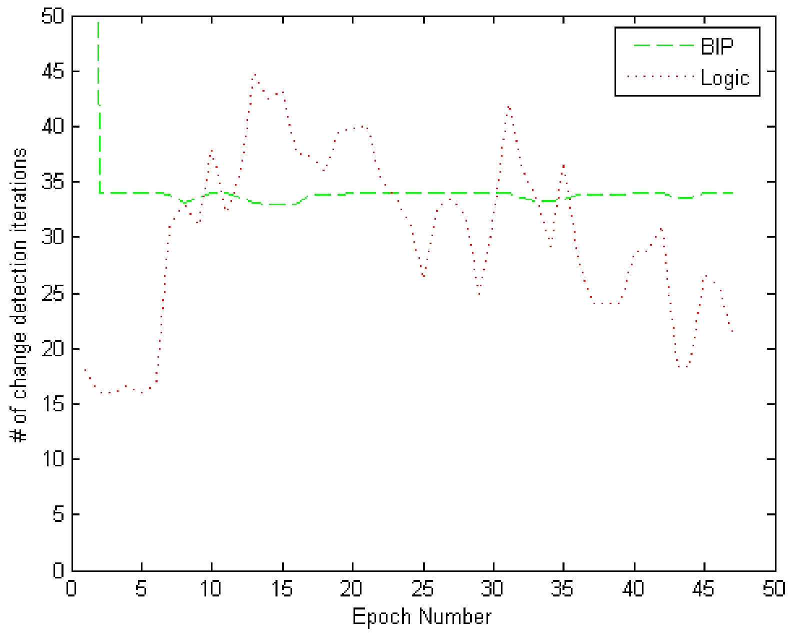


Figure 4.9 Scenario 1: Average change detection iterations for BIP and Logic, by epoch

is defined as:

$$\Psi = \sum_{j=1}^{3} \tau_j v_j \tag{4.9}$$

This will be used to determine the relative cost for the simulations.

Using the Monte Carlo simulation described in Section 4.1.8, the following tables show the differences in computations required to complete the operations. Of particular interest are the bottom rows of Tables 4.2 and 4.3. These rows indicate the amount of savings generated by choosing to perform the resource allocation by BIP instead of Logic. As can be seen, BIP uses far fewer texture and color measurement operations than Logic. BIP does use more mass operations. This is a result of the higher number of change detections that BIP is allowed to do through the constraint functions. By performing more mass operations, BIP is able to ensure that there is a wider area of the image that is consistently checked for new objects entering the image and at finer detail through the use of higher-resolution sub-regions. Overall, the BIP provides a net savings in computational tokens of 21.2% and 8.7% for scenarios 1 and 2 respectively. It must be recognized that

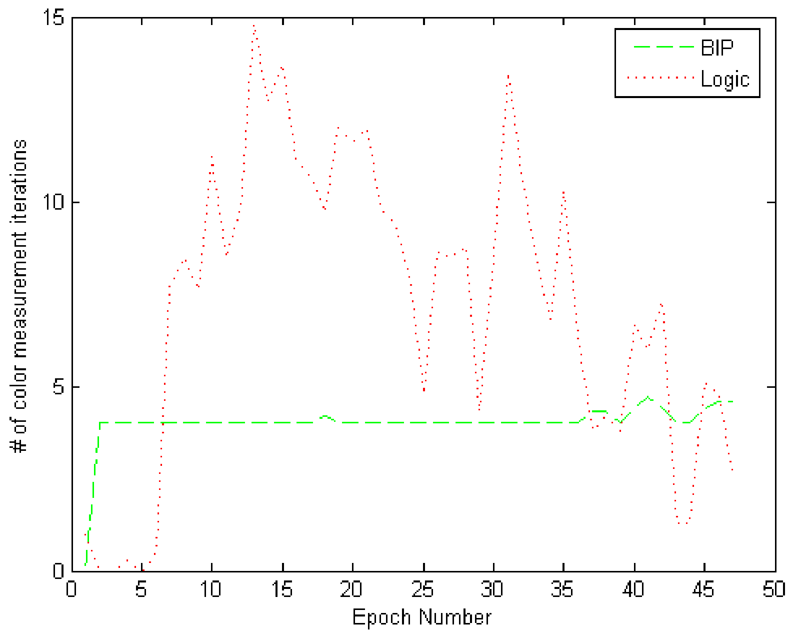


Figure 4.10 Scenario 1: Average color measurement iterations for BIP and Logic, by epoch

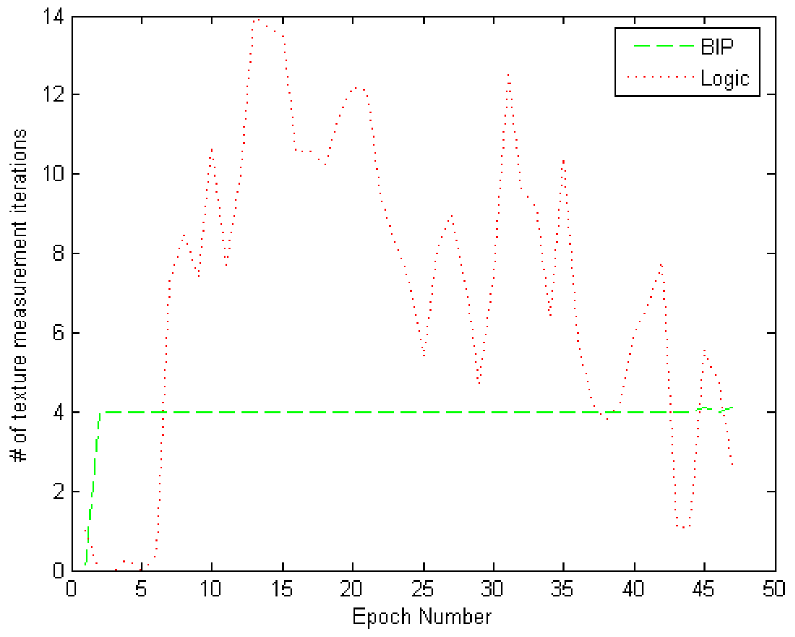


Figure 4.11 Scenario 1: Average texture measurement iterations for BIP and Logic, by epoch

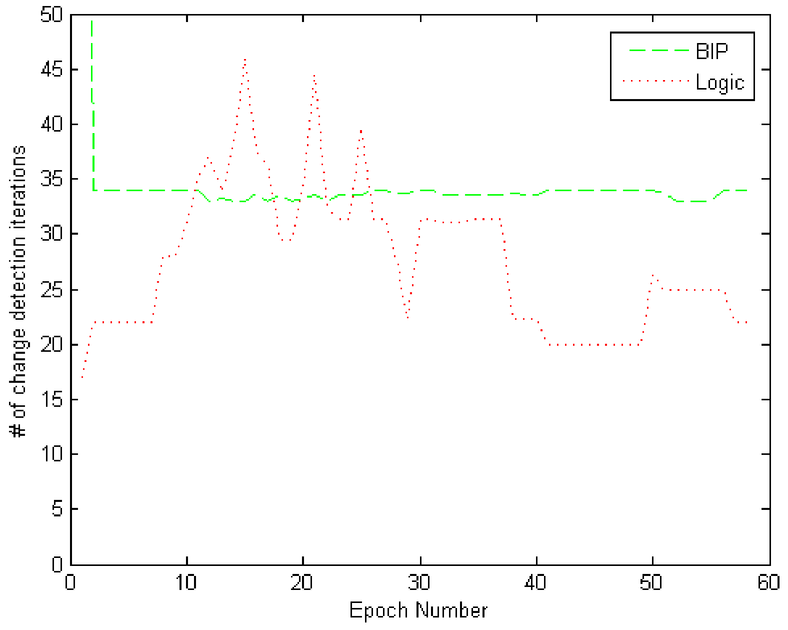


Figure 4.12 Scenario 2: Average change detection iterations for BIP and Logic, by epoch

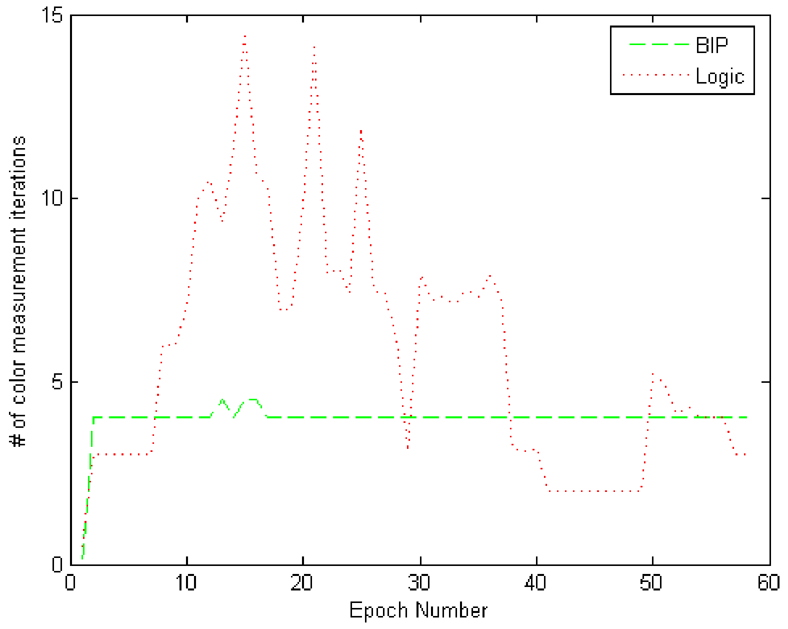


Figure 4.13 Scenario 2: Average color measurement iterations for BIP and Logic, by epoch

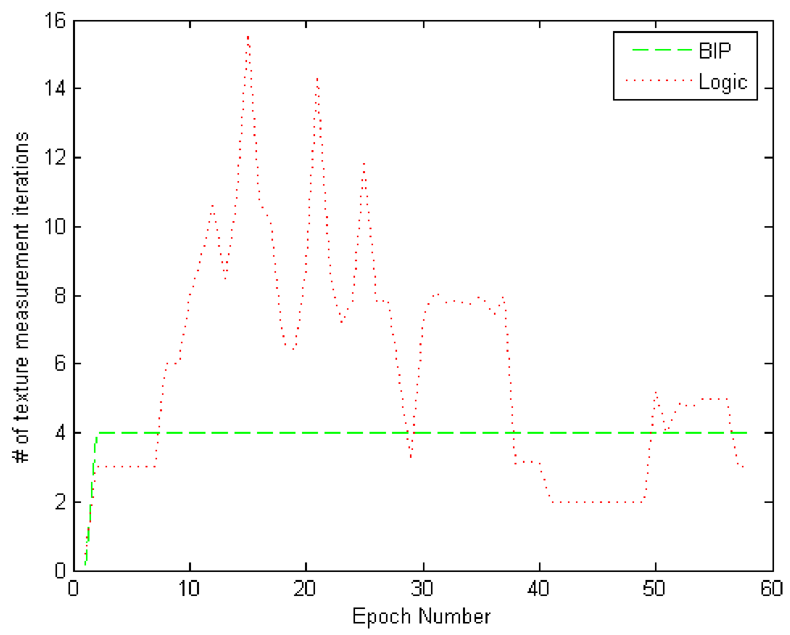


Figure 4.14 Scenario 2: Average texture measurement iterations for BIP and Logic, by epoch

the computational load of implementing the BIP algorithm is *not* included here. This comparison is a subject of future research.

4.2.2 Classification Accuracy. Accuracy of target classification is another important measure of the resource allocation effectiveness. This measure is based on how often the result of the classification algorithm is a false or true identification. The trends in the classification as observed over the multiple Monte Carlo runs will also be explored in order to determine which conditions lead to one classification or the other being more common. Runs which have unique characteristics will be displayed and analyzed for classification accuracy. For these scenarios, the target classifications will be named in accordance with the definitions provided at the beginning of this chapter. For Figures 4.15 through 4.43, blue indicates a "Decoy" classification, green indicates "Unknown" and red indicates a "Desired" classification. The shape used to indicate a target location in these figures is inconsequential.

4.2.2.1 Scenario 1. In the first scenario, the desired target and a decoy travel side-by-side for several frames, partially merging together at times. This behavior causes their change masks to overlap partially, and the result is they are treated as one target for several frames. More advanced image processing techniques may be able to separate the targets. However, these techniques are not available to the computer for this demonstration; so the targets' information will be frequently mixed during this time period for all of the runs. This leads to varied classification results and frequently changing classifications. Other features of Scenario 1 are that two targets are occluded by a structure during portions of their path traversal. Also, all of the targets depart the scene without any nearby neighbors, allowing for a few frames wherein unhindered tracking and classification may be performed.

Run 1 has one of the more optimistic sets of BIP results, and Logic has classification results that are not so promising. Using BIP, (Figure 4.15) no declarations of target type are made until the targets emerge from an occlusion (See A in Figure 4.15). In this area, the desired target is misclassified as a decoy target. However, after a few more frames elapse, the target is reclassified as a desired target before entering its second occlusion (B). When the target emerges from the second occlusion, it is classified as the desired target and maintains that classification until the end of the scenario (C). The other two targets gather measurements, and there are a few false positives mixed with true negatives towards the end of the scenario (D, E). In the Logic version of this run, (see Figure 4.16) there are a few false positives towards the end of the run for both of the decoy tracks (A, B). This serves to show that the false positives generated by the BIP may not be caused by the resource allocation but rather because of tracking issues such as global track maintenance (discussed later). It may also be that the decoy target's attributes may be close enough to the desired target to receive enough "Yes" votes for a "desired" classification. However, there is a series of true positives towards the end of the BIP run (See C in Figure 4.15) that are registered as false negatives by Logic (C in Figure 4.16). The second run for this scenario shows results that are similar to the first run for the Logic allocator, but very different for the BIP allocator. In this run, shown in Figure 4.17, the BIP allocator classifies most of the targets as decoy targets (A, B). In Figure 4.18, the Logic algorithm results indicate that it gets many of the classifications correct.

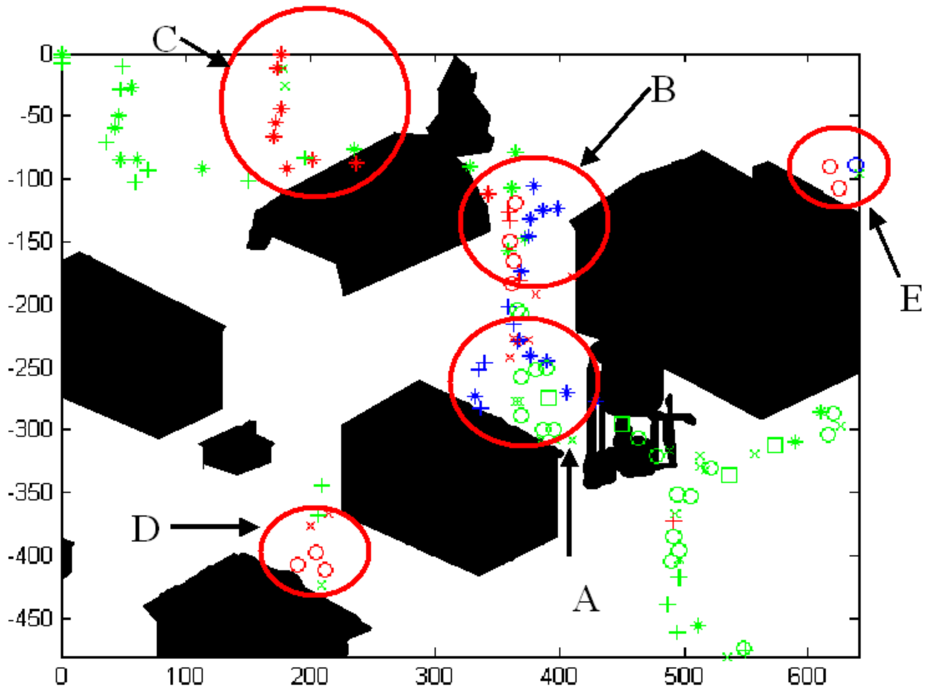


Figure 4.15 Scenario 1 BIP Run 1

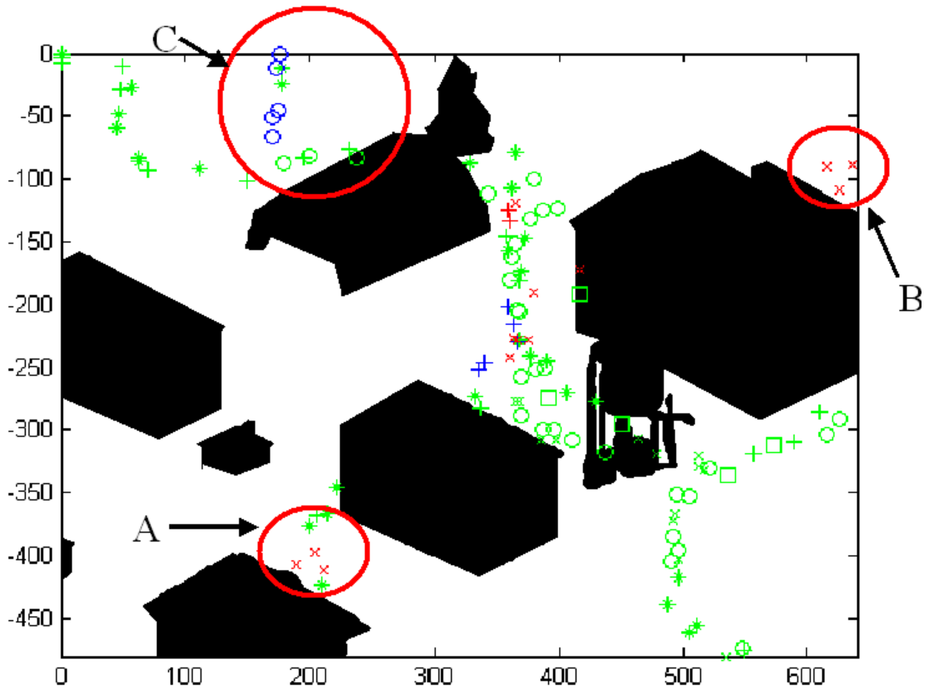


Figure 4.16 Scenario 1 Logic Run 1

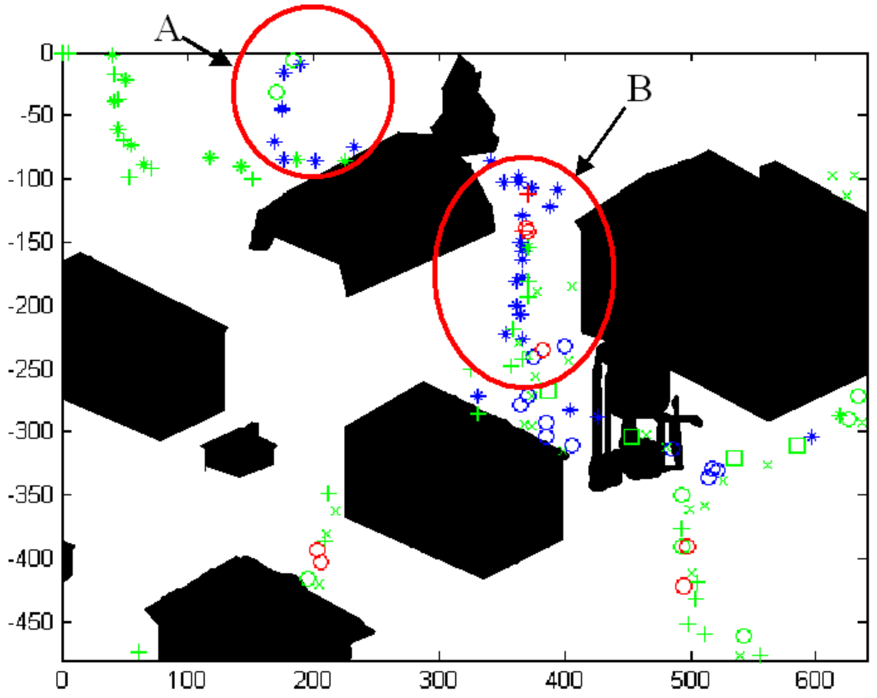


Figure 4.17 Scenario 1 BIP Run 2

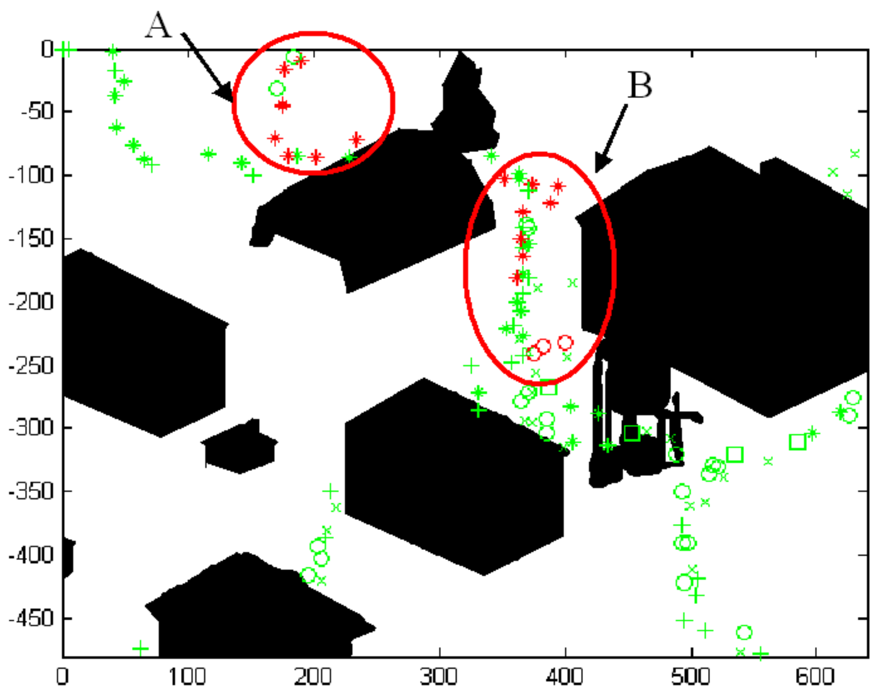


Figure 4.18 Scenario 1 Logic Run 2

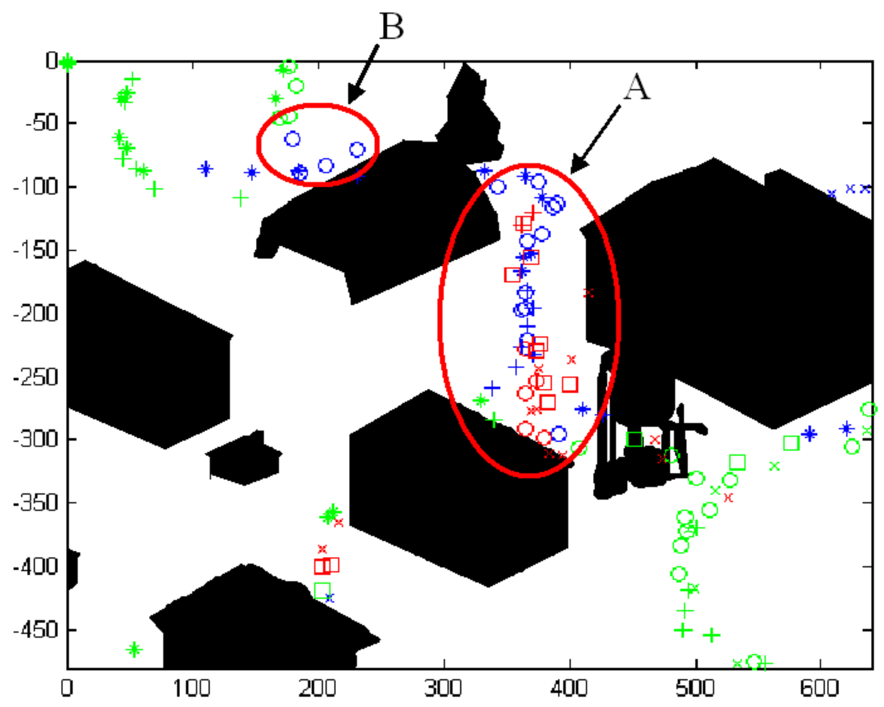


Figure 4.19 Scenario 1 BIP Run 3

Run 3 (See Figures 4.19 and 4.20) shows results through the center part of the image that are rather typical of most of the runs. As shown in Figure 4.19, the BIP allocator returns a trail of decoy classifications for the desired target in regions (A, B). The Logic allocator (Figure 4.20) returns results that are all unknowns, with a handful of false positives (A, B). This indicates that both algorithms are struggling due to the targets overlapping, creating a difficulty with proper classification.

Run 4 (not shown) shows trends that are similar to Run 3 for the BIP case. The major exception is that the decoy tracks do not gain enough votes to be given a desired classification, and are not measured as false positives. The Logic algorithm provided very similar results to those of Run 3 as well.

Run 5 exhibited some of the same trends as before. The BIP, shown in Figure 4.21, was again confused during the middle of the scenario by the overlapping targets (A). The last set of measurements were incorrectly classified using BIP (B). The Logic algorithm, Figure 4.22, maintains an unknown state for all targets during the overlap (B), and then

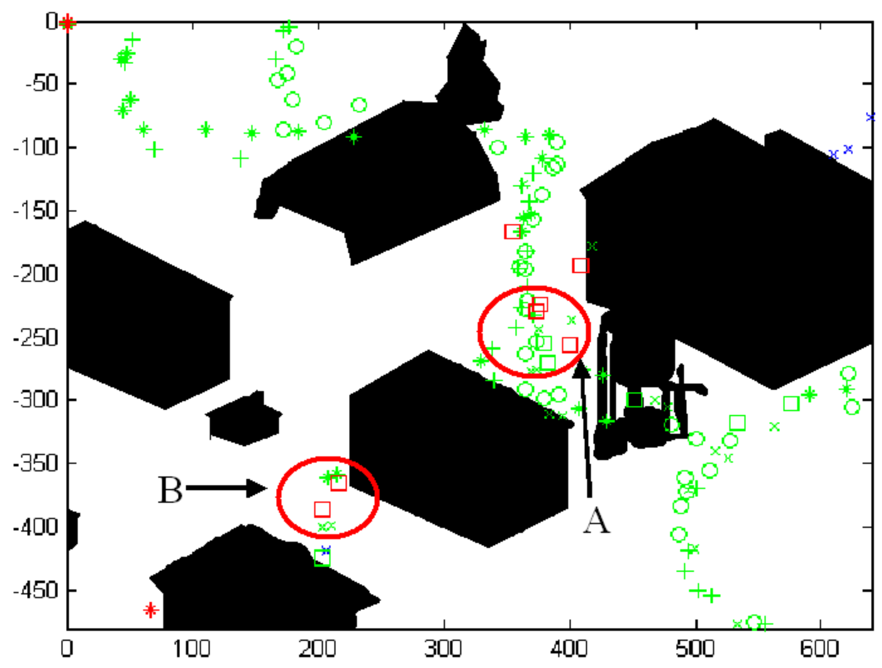


Figure 4.20 Scenario 1 Logic Run 3

classifies the targets correctly once they have moved away from each other (A, C, D). Notably, it correctly classified the desired target at the end of its path (A).

Run 6 results (see Figure 4.23) are almost identical to Run 1, with the exception that the targets are all classified as decoys during the overlapping epochs of the BIP case (A). However, with the exception of a few false positives, the targets are correctly classified towards the end of the scenario. As with Run 1, there are very few true positives using Logic, shown in Figure 4.24. Most of the classifications are unknown or false positives (A, B).

Run 7 generates an interesting trend during the BIP run that can be seen in Figure 4.25. Although the total number of true positives is similar to the previous runs, the trends are different. This run shows that depending on the allocations made by BIP, the target information may be extracted properly, as there are several true positives registered when the targets are together (A, B). In this case, the BIP allocator did not allocate resources to the sub-region containing the decoy target. In previous runs, incorrect classification resulted when resources were allocated to a decoy that was close to the desired target. The

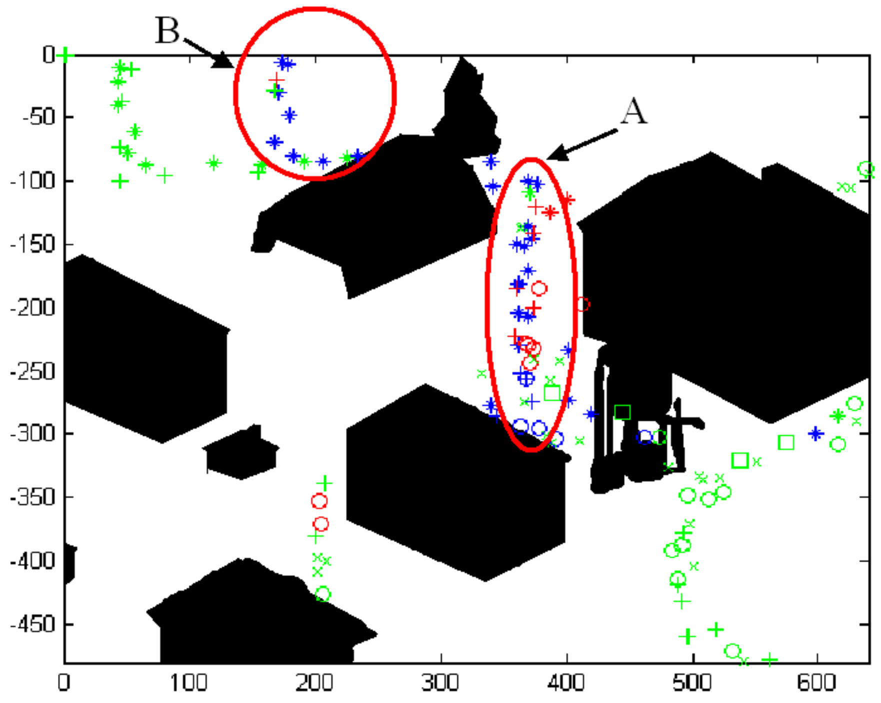


Figure 4.21 Scenario 1 BIP Run 5

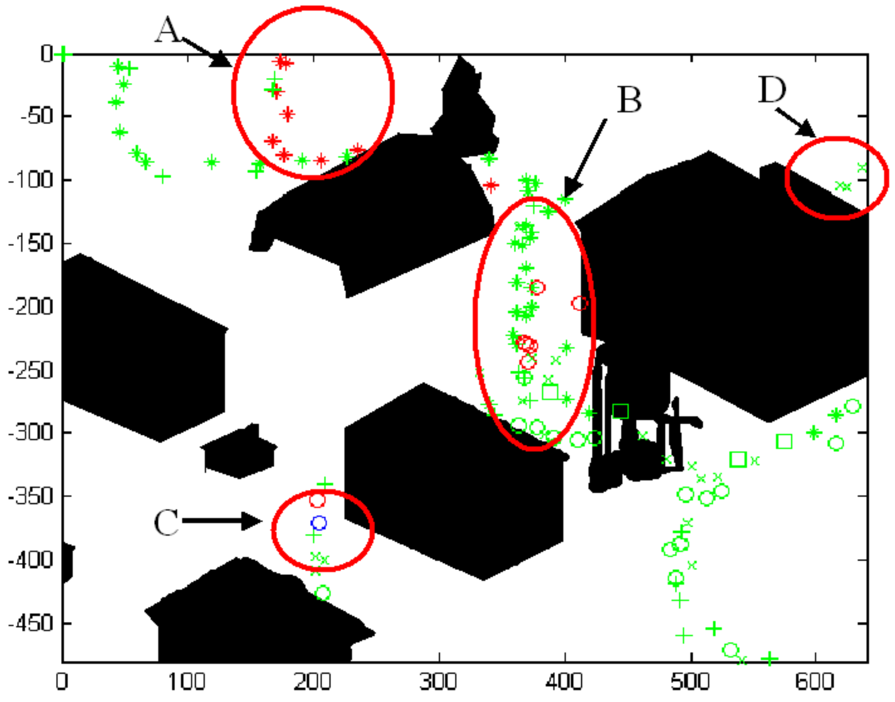


Figure 4.22 Scenario 1 Logic Run 5

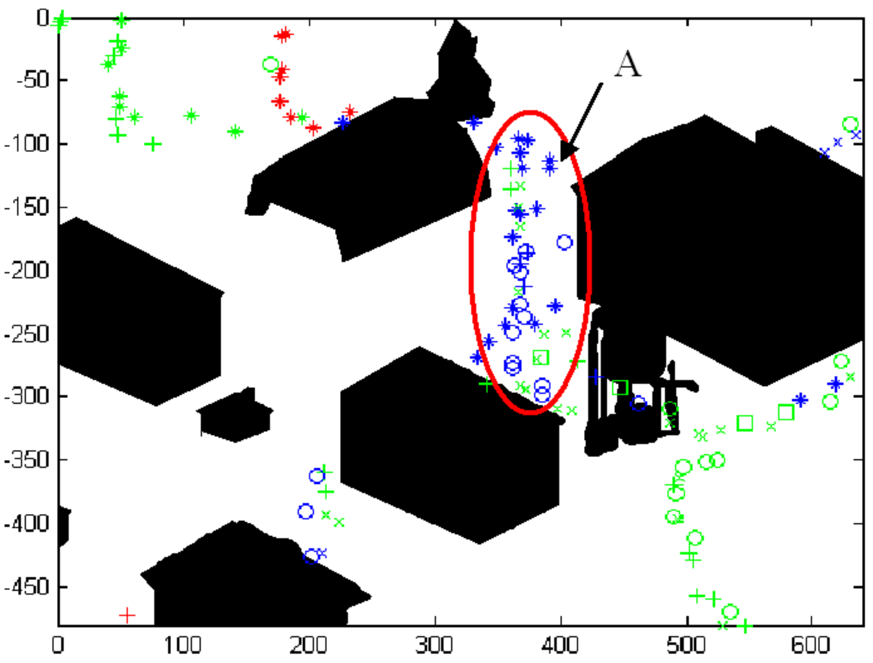


Figure 4.23 Scenario 1 BIP Run 6

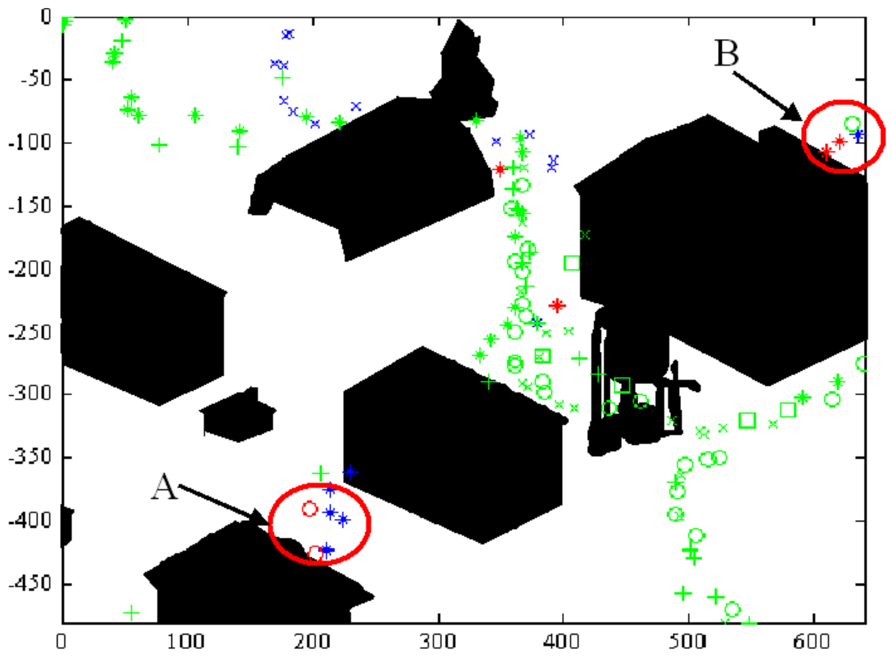


Figure 4.24 Scenario 1 Logic Run 6

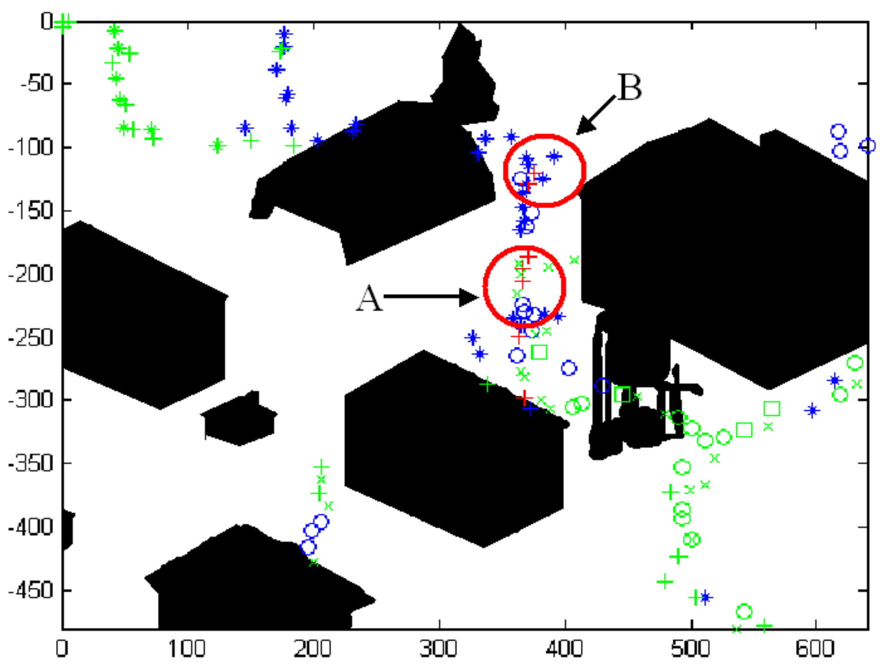


Figure 4.25 Scenario 1 BIP Run 7

results from Logic for this run are almost identical to Run 9 and are not shown separately.

Run 8 shows very good results for the BIP allocator (Figure 4.26). Specifically, while there are no positive classifications in the early epochs, many true positive classifications are made in the later epochs. They occur both through the overlapping period and after the final occlusion. The Logic allocator (Figure 4.27) shows results similar to Run 3. There are several false positives indicated in this run as well (B). As before, most of them occur at the end of the decoys' runs (A).

The results of Run 9 show very few positive classifications by BIP (see Figure 4.28). There are two at the end of the run (A) and a few false positives for the decoys (B). The Logic run, Figure 4.29, fails to classify any targets as either desired or decoy, with two exceptions (A). Almost every classification is to the unknown state. From this, it seems that there was never a case in which enough votes were applied towards any target to move it into either of the definite classifications.

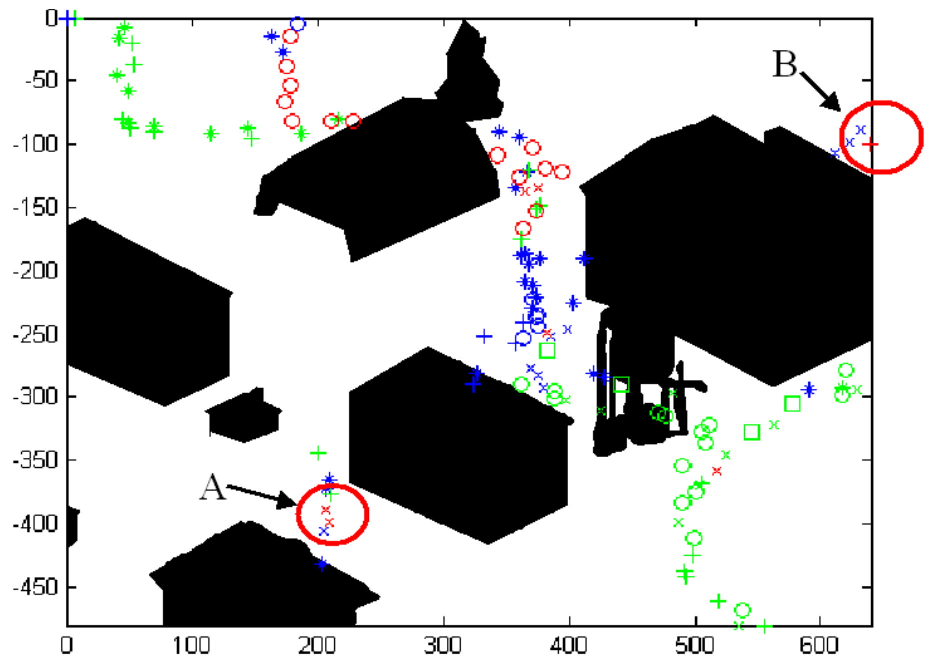


Figure 4.26 Scenario 1 BIP Run 8

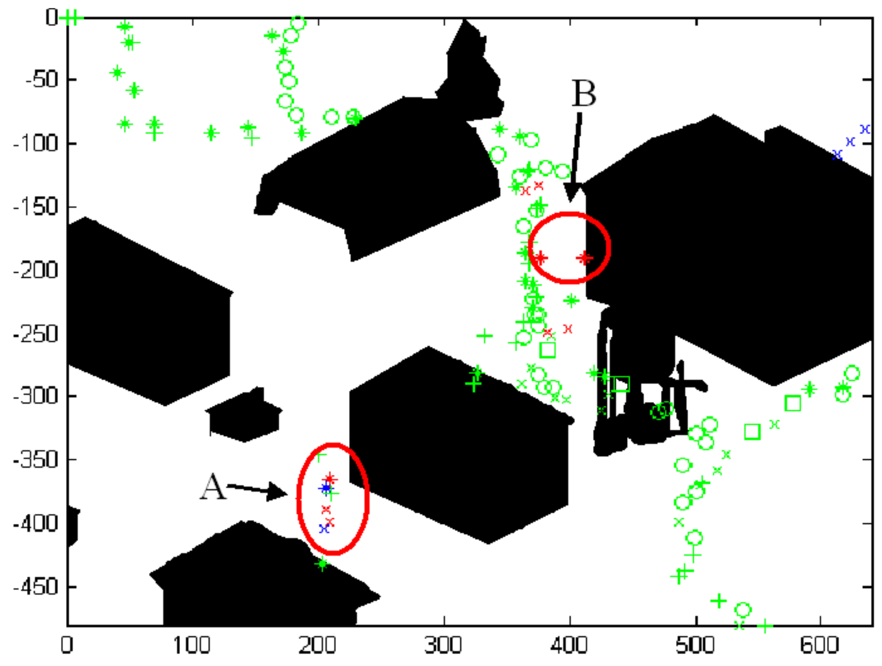


Figure 4.27 Scenario 1 Logic Run 8

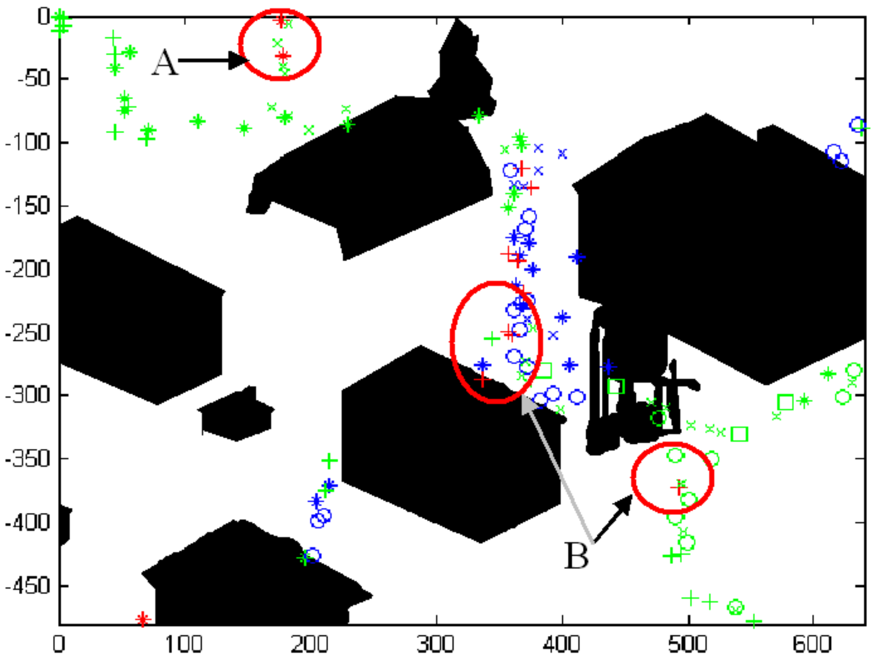


Figure 4.28 Scenario 1 BIP Run 9

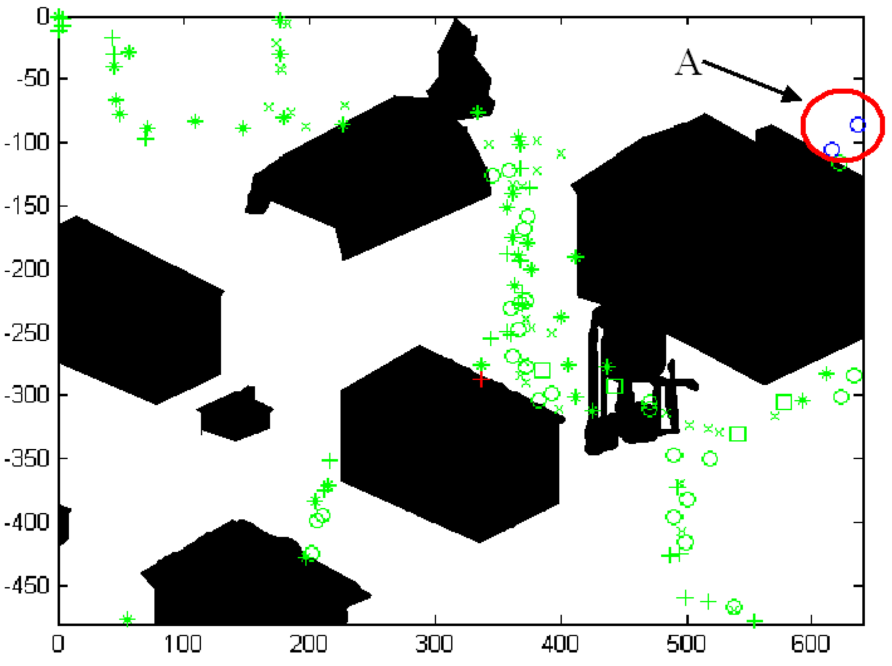


Figure 4.29 Scenario 1 Logic Run 9

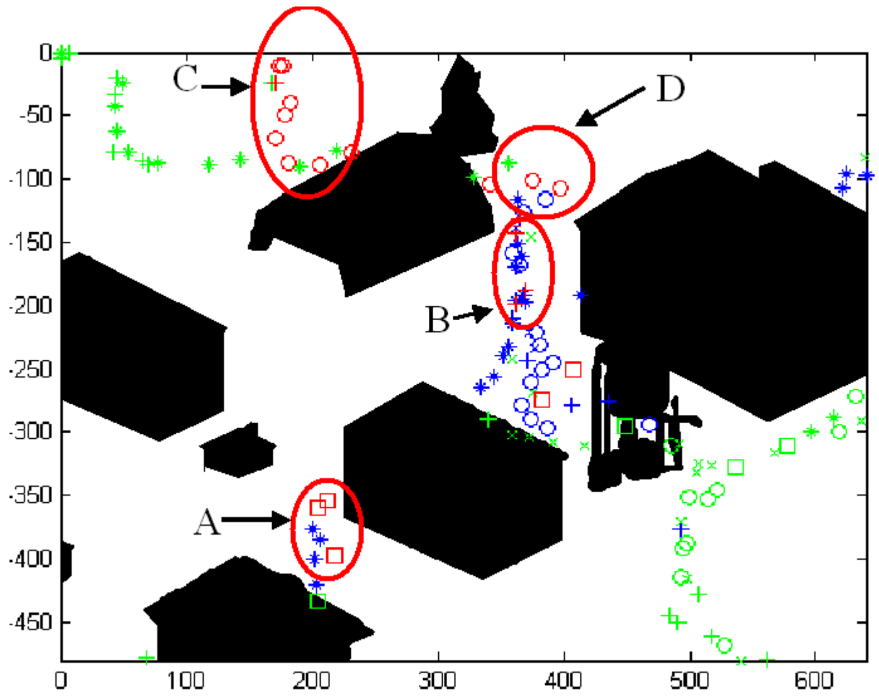


Figure 4.30 Scenario 1 BIP Run 10

The final run, Run 10, indicates some of the best sets of results for this scenario using either algorithm. Figures 4.30 and 4.31 show these results. While both algorithms have a high set of false positives (A, B), both of them also have a high number of true positives (C, D). There are also a high number of decoy classifications for both algorithms. This is a promising result, as it indicates that the algorithms were able to use the available attributes to distinguish the decoys from the desired target.

Table 4.4 shows the numbers of true and false positive and negative classifications for each of the runs in Scenario 1. Overall, there are 30 opportunities for a desired identification in each run, and 48 opportunities for a decoy identification. The bottom row shows conditional probabilities associated with the classification accuracies, and will be described in detail in Section 4.2.3.

4.2.2.2 Scenario 2. Scenario 2 does not have a case in which targets directly overlap each other; however, there are two events that involve targets crossing paths. There is also an idling event that is shown as a cluster of measurements in the bottom left corner

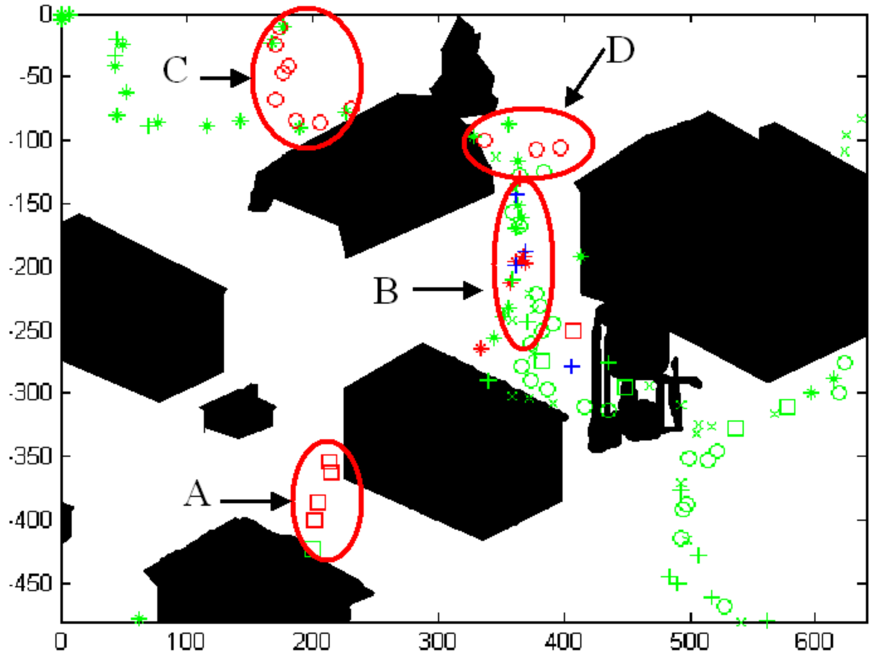


Figure 4.31 Scenario 1 Logic Run 10

Table 4.4 Scenario 1 Classifications

	BIP				Logic			
	True		False		True		False	
Run	Pos.	Neg.	Pos.	Neg.	Pos.	Neg.	Pos.	Neg.
1	13	39	9	17	3	40	8	27
2	3	40	8	27	15	45	3	15
3	4	38	10	26	1	42	6	29
4	7	47	1	23	0	44	4	30
5	6	40	8	24	10	45	3	20
6	6	47	1	24	1	43	5	29
7	5	47	1	25	0	48	0	30
8	12	43	5	18	3	43	5	27
9	5	44	4	24	0	48	0	30
10	12	41	7	18	10	39	9	20
Total	73	426	54	227	43	435	45	257
	$C_D\|T_D$	$C_N\|T_N$	$C_D\|T_N$	$C_N\|T_D$	$C_D\|T_D$	$C_N\|T_N$	$C_D\|T_N$	$C_N\|T_D$

of all of the plots. A good classification trend is present at the end of the desired target's path. After it emerges from the final occlusion and approaches the right edge of the screen, most of the runs show that the target is classified correctly. Overall, this scenario is more sparsely populated than the previous scenario. This means that, on average, a given target will have more resources available to it when using BIP, as there will not be a need to divide resources as much between the other targets. In this scenario there are 27 opportunities for a desired track classification and 45 opportunities for a decoy classification.

In the first run of the second scenario (see Figures 4.32 and 4.33), several trends can be observed that are common through the rest of the runs. For most of the runs, the desired target is classified correctly as it nears the occlusion at the top left of the screen (see A in both figures). Also, in most cases the desired target does not obtain a true positive during its transition from the first occlusion to the occlusion in the bottom center of the screen (B). The desired target typically receives a false negative in this region. Both algorithms correctly classify the idle target consistently (C), and no incorrect classifications are made on that target until it begins moving. Finally, a trend that is present in most of the Logic runs is a small number of true positives for the desired target, and no false positives for the decoys. The only time the desired target is classified correctly is for the last five epochs before exiting the scene (Figure 4.33 D). In contrast, the BIP algorithm generates many more true positives during various segments of the desired target's path.

The results of BIP for Run 2, (Figure 4.34), show that the desired target was also classified correctly long before entering its first occlusion (A). However, this classification was not retained when it re-emerged from the occlusion (B). Run 2 has results that are unique to Logic for these sets of runs (see Figure 4.35). Logic shows many true positives as the desired target enters the first occlusion (A), and several false positives as one of the decoy targets enters the final occlusion (B).

For Run 3, the BIP showed some differing results, (Figure 4.36). Using BIP, there were more true positive classifications than for Run 1 and more false positive classifications than for Run 2 (A, B, C). Using BIP, the results for Run 4 (not shown) are very similar in form to the results in Run 2. The results for runs 3 through 7 for the Logic algorithm were identical to the first run, Figure 4.33, and are not shown seperately.

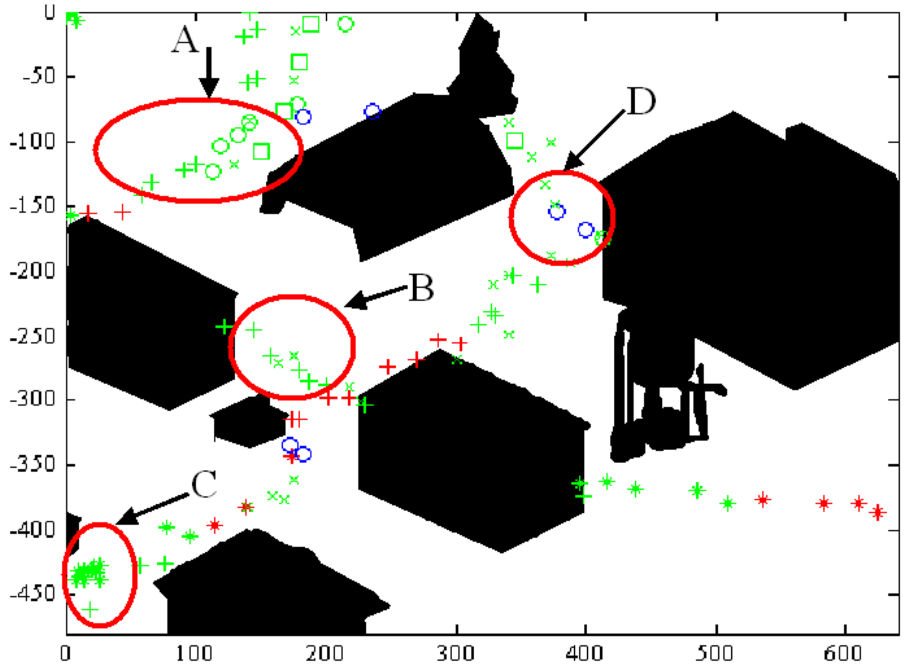


Figure 4.32 Scenario 2 BIP Run 1

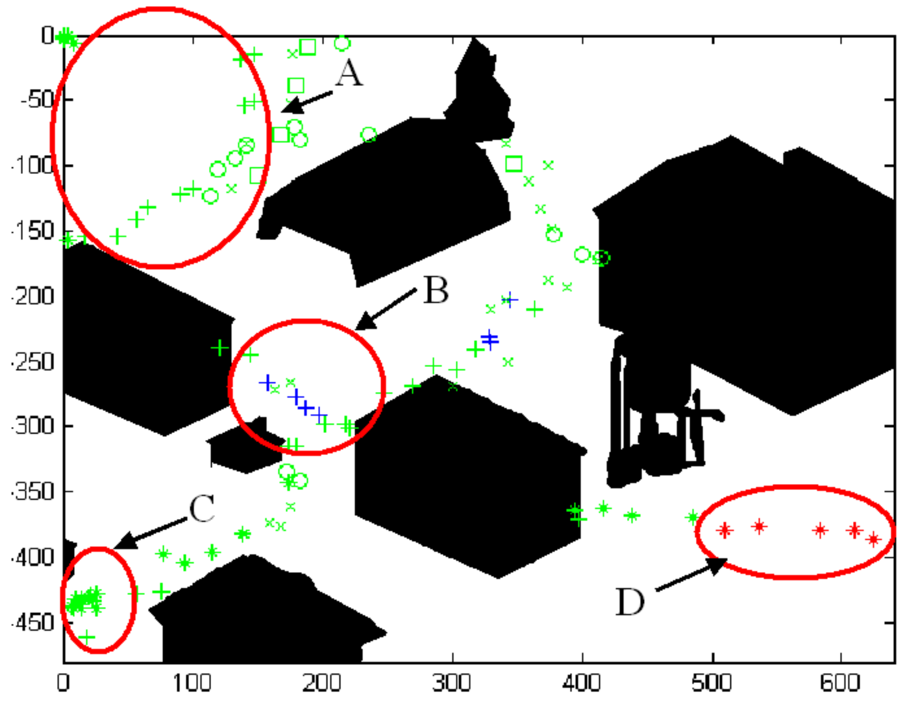


Figure 4.33 Scenario 2 Logic Run 1

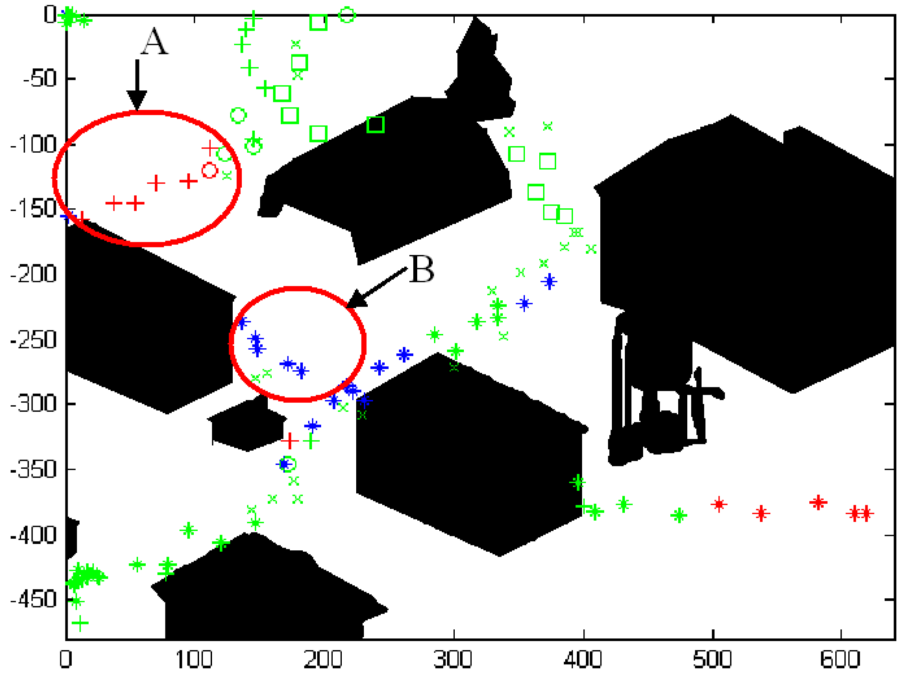


Figure 4.34 Scenario 2 BIP Run 2

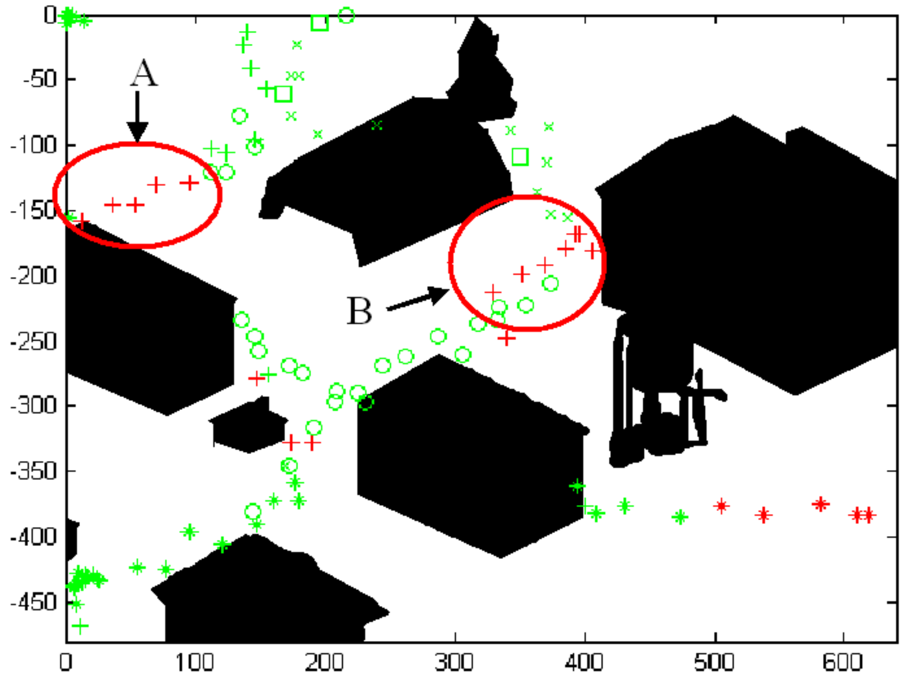


Figure 4.35 Scenario 2 Logic Run 2

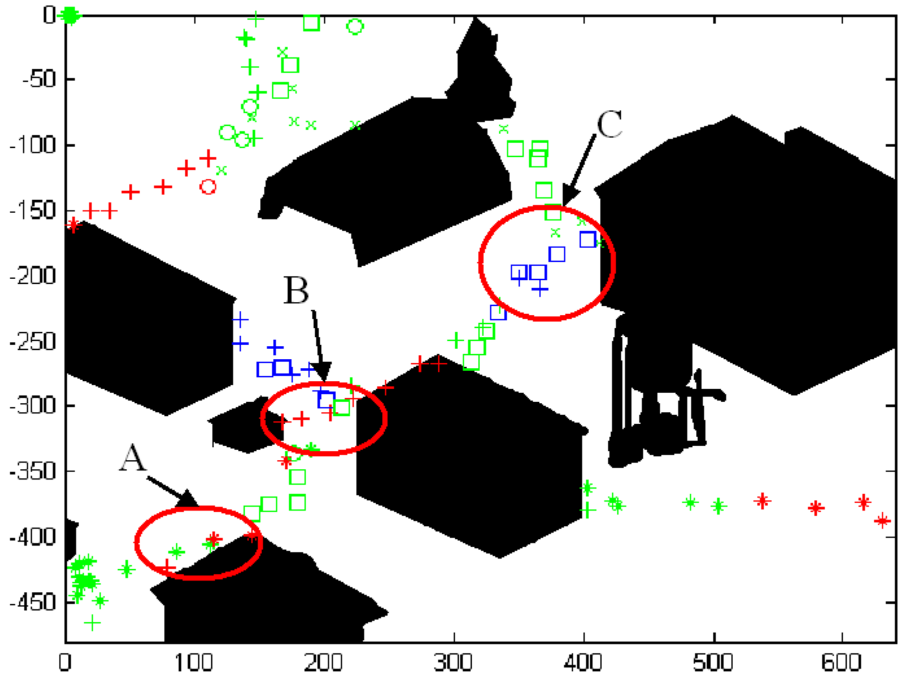


Figure 4.36 Scenario 2 BIP Run 3

Run 5 using BIP shows an interesting change from the previous results. In this run (Figure 4.37) the algorithm applies resources to the desired target as it moves between its first and second occlusion (A). As a result, it is correctly classified during this period. However, the algorithm also incorrectly classifies one of the decoys (B).

Run 6 shows nothing special or unique using either algorithm. Run 7 (see Figure 4.38) has an interesting result while the desired target is between the first and second occlusions. Both the desired target and a decoy seem to change from one classification to the other, (A, B in Figure 4.38). It is not readily apparent what causes this change, but it could be the result of a global track maintenance error or caused by an error in target association.

Run 8 shows new results using both allocators. The BIP results in Figure 4.39 are a combination of patterns that have been observed in other runs. There are a series of false positive classifications from the target that moved from the bottom left towards the top-right part of the screen (A). However, there are no correct classifications of the desired target as it moves between the two occlusion regions (B). BIP did correctly classify the

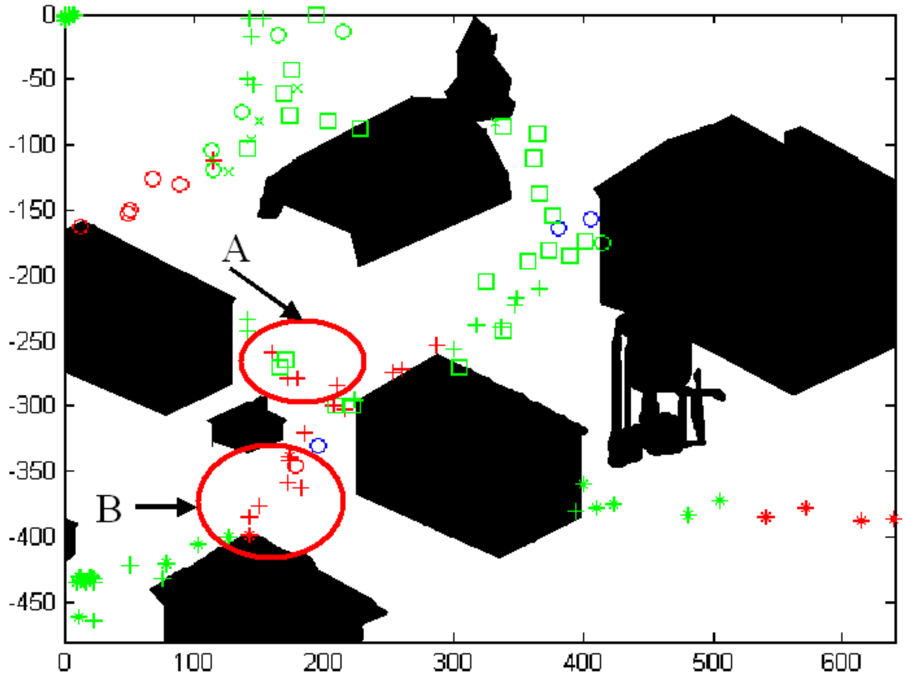


Figure 4.37 Scenario 2 BIP Run 5

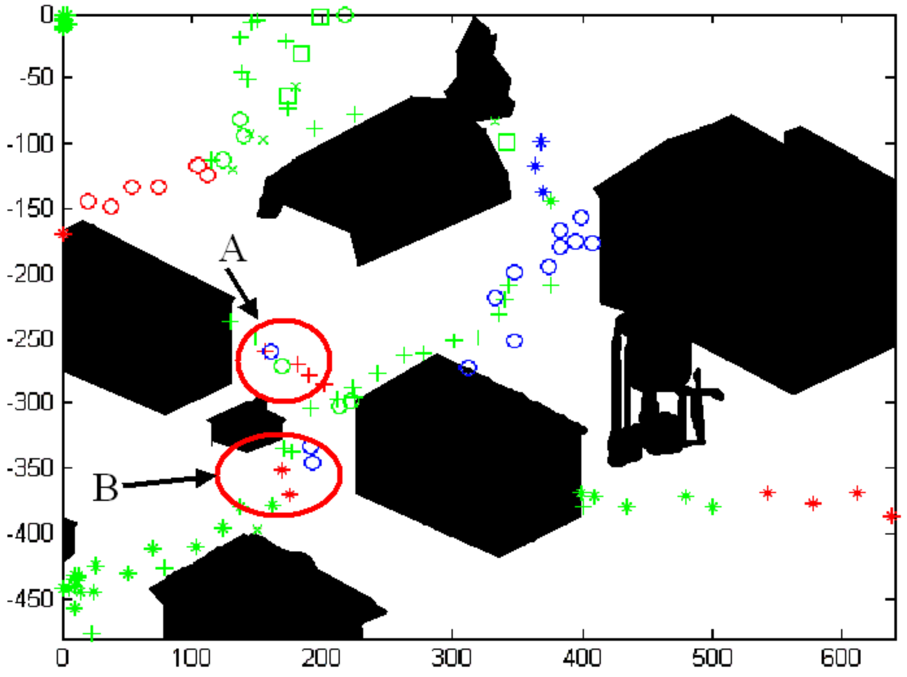


Figure 4.38 Scenario 2 BIP Run 7

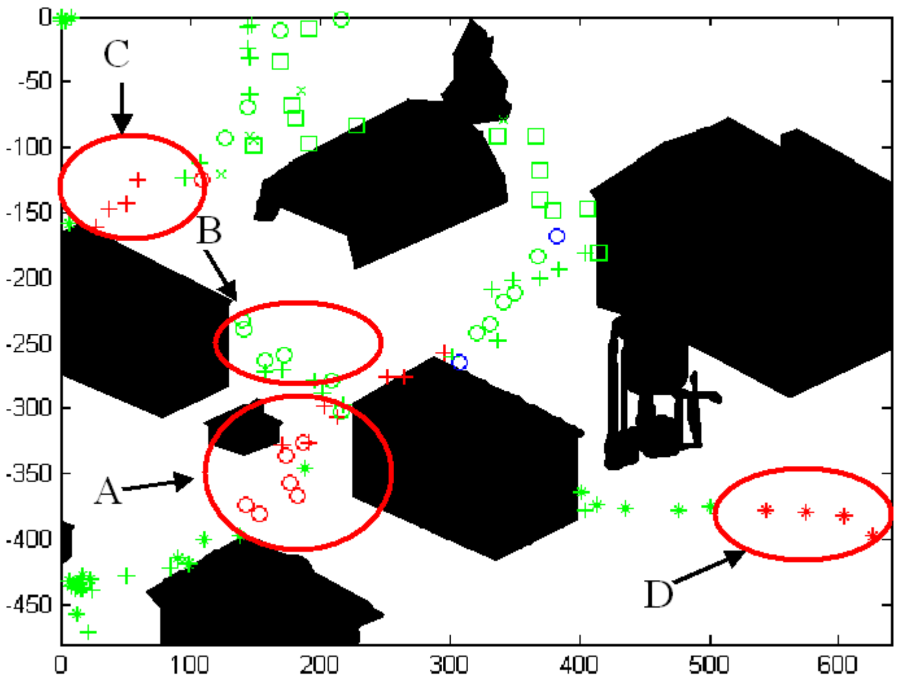


Figure 4.39 Scenario 2 BIP Run 8

desired target as it enters the first occlusion (D) and towards the end of the run (C). Likewise, the Logic allocator (Figure 4.40) produced a series of false positives as a decoy target moves through the middle of the scene (A) and correctly identified the desired target at the end (B).

The BIP results for Run 9 are very similar to the BIP results from Run 8. The Logic algorithm, (see Figure 4.41), correctly classifies the desired target after exiting the first occlusion region (A). Also, as with Run 2, there were a significant number of incorrect classifications of the decoys that hinder the performance of this algorithm (B).

BIP provides relatively poor performance in Run 10. Figure 4.42 shows several false positives (A) and few true positives (B). The Logic allocator (Figure 4.43) behaves almost identically to Run 2, except that there are fewer false positives (A). Table 4.5 provides a summary of the performance for both algorithms for Scenario 2.

4.2.2.3 Mutual Trends. In general, both of the algorithms were much more likely to make negative classifications than positive classifications. This is likely

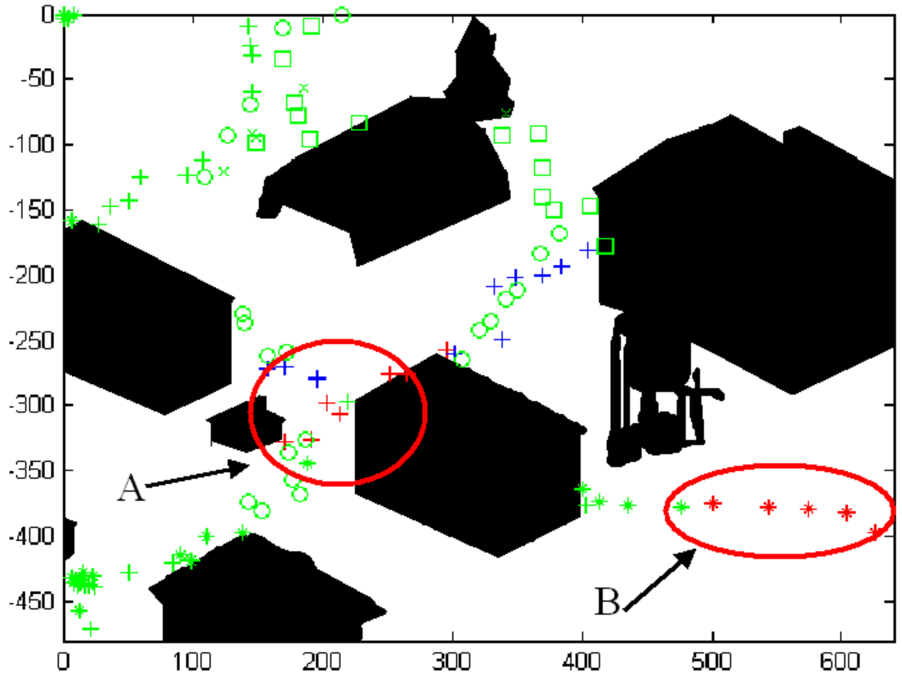


Figure 4.40 Scenario 2 Logic Run 8

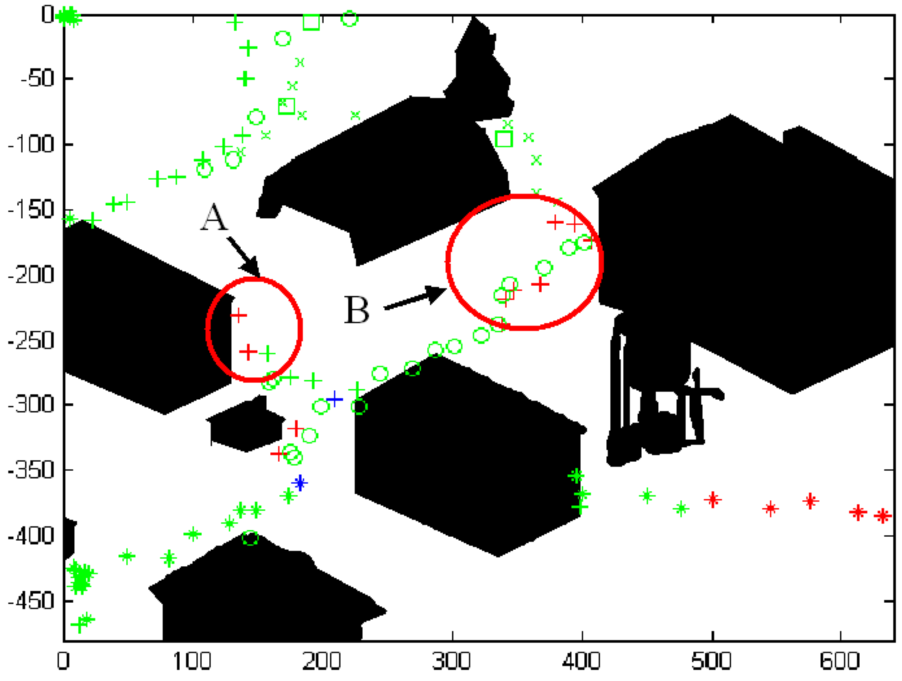


Figure 4.41 Scenario 2 Logic Run 9

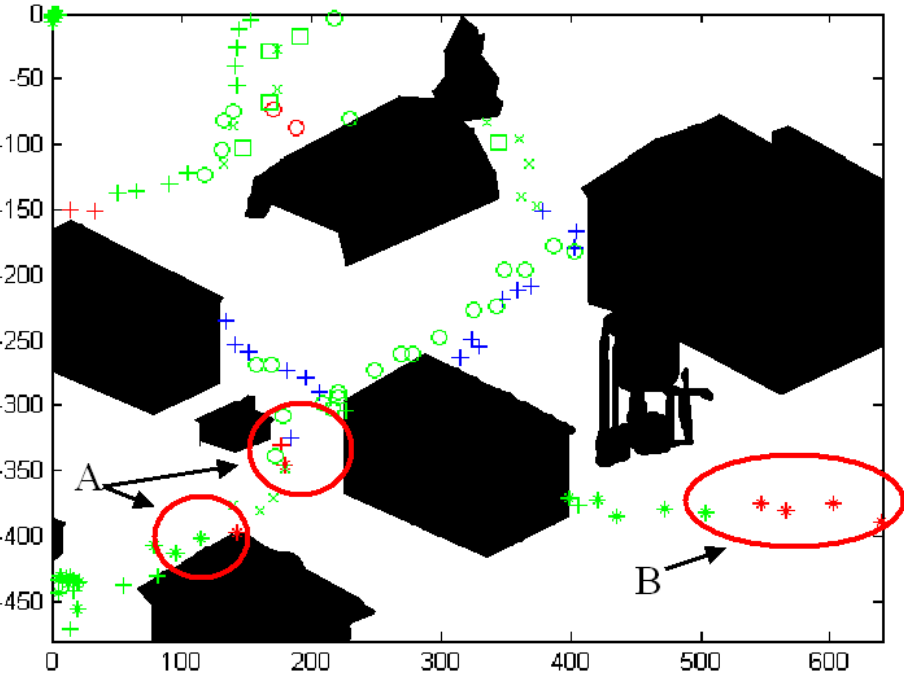


Figure 4.42 Scenario 2 BIP Run 10

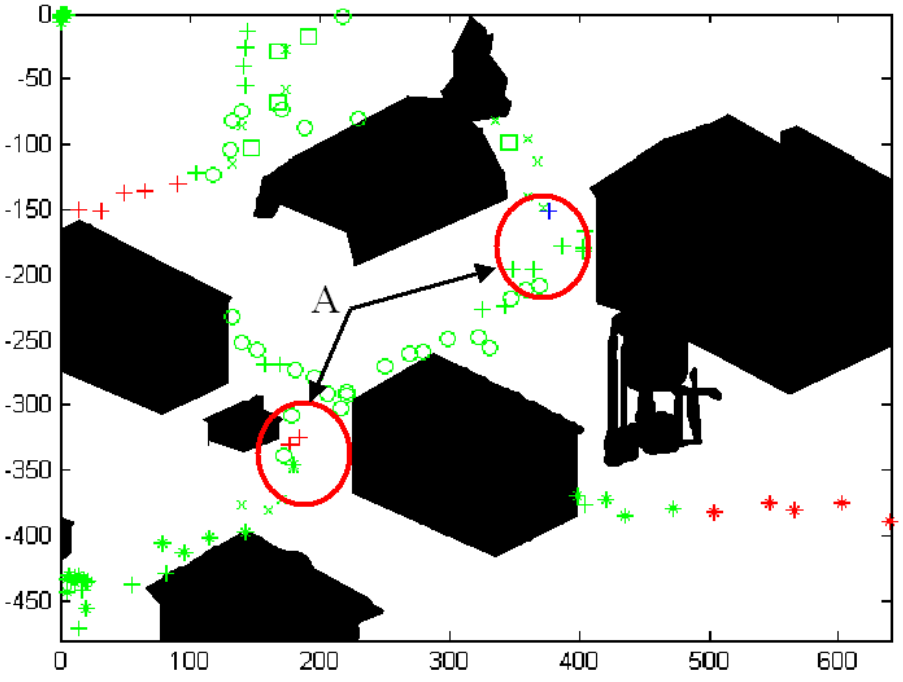


Figure 4.43 Scenario 2 Logic Run 10

Table 4.5 Scenario 2 Classifications

	BIP				Logic			
	True		False		True		False	
Run	Pos.	Neg.	Pos.	Neg.	Pos.	Neg.	Pos.	Neg.
1	6	36	9	21	5	45	0	22
2	12	44	1	15	11	36	9	16
3	11	37	8	16	5	45	0	22
4	9	45	0	18	5	45	0	22
5	13	34	11	14	5	45	0	22
6	10	41	4	17	5	45	0	22
7	14	43	2	13	5	45	0	22
8	9	35	10	18	5	39	6	22
9	10	36	9	17	7	36	9	20
10	6	40	5	21	10	43	2	17
Total	100	391	59	170	63	424	26	207
	$C_D\|T_D$	$C_N\|T_N$	$C_D\|T_N$	$C_N\|T_D$	$C_D\|T_D$	$C_N\|T_N$	$C_D\|T_N$	$C_N\|T_D$

related to the values in the confusion matrix and characteristics of the voting algorithm. Another apparent trend included targets that were "alone" in open terrain being typically classified properly, usually between three and six epochs. However, the decoy targets were not typically classified as decoys, but rather classification remained in the transitory "unknown" state. This results because the target would have to receive either zero or one votes in order to begin to be classified as a decoy. Since the dismount targets will usually be of about the same size, and obtain one vote for mass, if a decoy target has another attribute that satisfies the requirements for a "yes" vote, then the target will be classified as an unknown target.

4.2.3 Classification Accuracy Summary. From the information presented in Tables 4.4 and 4.5, it is possible to determine the probability of a target being of one of the two types, desired or decoy, given a classification. There are four permutations that can be measured. These are the probabilities of having desired or decoy targets, given that there is a positive or negative classification. Conditional probabilities can be calculated via:

$$P(T_x|C_y) = \frac{P(C_y|T_x)P(T_x)}{P(C_y)} \tag{4.10}$$

Table 4.6 Overall Classification Accuracy

	Scenario 1			Scenario 2		
	BIP	Logic	Best	BIP	Logic	Best
$P(T_D\|C_D)$	**0.575**	0.500	BIP	0.629	**0.708**	Logic
$P(T_N\|C_D)$	**0.425**	0.500	BIP	0.371	**0.292**	Logic
$P(T_D\|C_N)$	**0.348**	0.37	BIP	**0.303**	0.328	BIP
$P(T_N\|C_N)$	**0.652**	0.63	BIP	**0.697**	0.672	BIP

where T_x indicates a target truly being desired or decoy and C_y indicates a desired or decoy classification.

The values in Table 4.6 indicate the probabilities given by Equation (4.10). The variable T_D indicates a (D)esired target, while T_N indicates the (N)on-desired or decoy target. The C_D indicates a (D)esired classification, while C_N indicates the (N)on-desired or decoy classification. The algorithm that provides the better performance for each case is indicated in the right column of each of the scenarios' information. The probabilities, $P(T_N)$ and $P(T_D)$ are determined by the total number of times each of the target types appears in the scene, divided by the total appearances of all targets. The probabilities $P(C_D)$ and $P(C_N)$ are the relative chance of each of the two classifications occurring, and are computed as finding the number of times each classification is made divided by the total number of classifications made. To find the four conditional probabilities, $P(C_y|T_x)$, the value in the bottom row of the two tables is divided by the sum of the values that correspond to that target type. For example, in Scenario 2, with BIP, the total number of non-desired targets is 450. Of those, 391 are given a decoy classification. The value for $P(C_N|T_N)$ is therefore $\frac{391}{450}$. Table 4.6 shows the full conditional analysis for each of the scenarios and resource allocation algorithms. The desired results are to have large values of $P(T_D|C_D)$ and $P(T_N|C_N)$ and small values for $P(T_N|C_D)$ and $P(T_D|C_N)$. The entries in bold show which results are better.

Two other valuable metrics of performance are the probability of detections and false alarms. Given the probability of detection, represented as P_D, is defined as the ratio of

Table 4.7 Detection and False Alarm Accuracy

	Scenario 1			Scenario 2	
	BIP	Logic		BIP	Logic
P_C	0.640	0.613		0.682	0.676
P_{FA}	0.360	0.387		0.318	0.324
P_C/P_{FA}	**1.776**	1.583		**2.144**	2.090

true positives and true negatives to total classifications:

$$P_C = \frac{C_D|T_D + C_N|T_N}{C_D|T_D + C_D|T_N + C_N|T_D + C_N|T_N} \tag{4.11}$$

Likewise the false alarm rate, P_{FA}, is defined as the ratio of false positives and false negatives to total classifications by:

$$P_{FA} = \frac{C_D|T_N + C_N|T_D}{C_D|T_D + C_D|T_N + C_N|T_D + C_N|T_N} \tag{4.12}$$

These probabilities are obtained from the number of events $C_x|T_x$ as tabulated in Tables 4.4 and 4.5. The ratio of detections to false alarms, P_D/P_{FA} is used to compare the performance between Logic and BIP for the two scenarios. Table 4.7 compares the two algorithms using these metrics, with the desire to have small values for P_{FA} and large values for P_D and P_D/P_{FA}.

4.3 Summary

This experiment shows that when comparing computational time expended, the BIP algorithm outperformed the Logic algorithm. The results varied depending on the amount of traffic present in the scenes; however, in both scenarios, the BIP obviously was able to use fewer resources. The experiment also showed some increase in classification performance through the use of BIP compared to the Logic algorithm. While the improvements were not tremendous, they do warrant further investigation. The next chapter will present more details as to the information gained through these results and present more ideas for further investigation.

V. Conclusions & Recommendations

5.1 Experimental Conclusions

The goal of this thesis was to present and demonstrate the effectiveness of an integer-program based method for determining the optimal resource allocation for an image processing problem. Through the use of a target-state prediction algorithm, the Kalman filter, a formulation was devised to populate the objective function of a binary integer program. Through the course of the experimentation, several important factors were observed regarding the use of high-dimensional binary integer programs. Formulations of the integer program were found that are very difficult to solve in a reasonable amount of time. Based on analysis of the BIP performance, the set of conditions necessary to obtain a quickly-solvable BIP were found to be a unimodular $\mathbf{A}$ matrix as defined by Equation (3.8). Furthermore, through an investigation of the results provided by the BIP, several important relationships were observed. As expected, the BIP chose sub-regions of the image that corresponded to the highest coefficients in the objective function. Also, the selections made by the BIP seemed to be rather intuitive. They consistently corresponded to areas of the image that satisfied one of the following conditions:

1. When a target was found via change detection, BIP allocated resources primarily towards those sub-regions.
2. When a target was occluded or not found via change detection, sub-regions immediately surrounding the expected location of the target, as defined by Kalman filter prediction, were routinely selected.
3. When many targets were found, sub-regions expected to contain the targets were examined, often at the expense of examining sub-regions adjacent to the targets, in which predicted locations were expected to be.

The results did defy intuition at some situations. One occasional event was when the objective function had uniform values for its coefficients. The fact that such an event could occur indicates an improperly formulated problem.

Another apparent problem was the result of an error in the global track maintenance and track management algorithms. These would be the result of using less sophisticated tracking algorithms that are more prone to error. In the figures shown in Chapter 4, some consistent erroneous track declarations are made. These include the BIP method consistently calling the target that is known to be the desired target, a decoy target during the transition between two of the occlusions. Another consistent error is when the Logic algorithm would refuse to make declarations regarding any track until the very end of the runs in Scenario 2. However, the BIP did perform as expected with regards to the handling of color and texture resource allocation. In the cases in which several targets were present, these more valuable resources were divided among the targets that had not received many of these measurement types. This seemed to help the algorithm obtain the important classification information, and played a large role in the overall performance of the system.

Overall, the Binary Integer Program (BIP) resource allocator seemed to perform better than the Logic resource allocator with both measures of performance, algorithm utilization and classification accuracy. Using the BIP resulted in a significant reduction in the number of iterations of each image processing algorithm. In Scenario 1, the BIP used 22% less time to perform image processing, and used 8.7% less time in Scenario 2. Using BIP also resulted in an improved accuracy in target classification. However, the comparison of detection versus false alarm rates produced ambiguous results. The results regarding the number of times the image processing algorithms were used does not account for the fact that the BIP is a much more computationally intensive method to implement than a simple logic algorithm. Although BIP still used fewer image processing resources than Logic, an informal side-by-side analysis on seemingly identical computers showed that Logic consistently performed its runs faster than BIP.

5.2 Recommendations

5.2.1 BIP Derivative Approaches. This thesis provides several areas for further investigation. More work should be done to formulate the objective function properly for the BIP. The value formulation appeared to be the greatest influencing factor on how the full algorithm performed. This is an intuitive result of the process, and indicates that the system is very responsive to the valuations assigned to a sub-region. A full study should include the various multipliers attached to the modifiers within the algorithm, and the method used to combine the weights from different targets. An improved method of sub-region valuation will assist any integer-programming based algorithm or a logic-based algorithm that assigns resources based on the relative weighting function.

Another approach, instead of improved sub-region valuation, is to do away with sub-region valuation altogether. Instead, a method could be devised wherein a target's expected state and covariance form the basis for a region of image processing. An advantage of this would be that such a technique would not be subject to the nuances of sub-region overlaps. This region could be sized depending on the needs of the particular target or constraints on the computational availability. Such an approach would most likely require the formulation of the resource allocator in a Linear Program style as described in Section 5.2.5. This is because the type of problem defined here does not meet the binary decision form of the BIP.

5.2.2 Additional Scenarios. The performance of the algorithm with other target types should be examined. This would include target groups and other moving objects such as vehicles and large shadows that would normally be encountered in urban environments. Testing on additional scenarios, and even real-world examples, would actually provide a better set of metrics for a proof-of-concept. Furthermore, testing on a real-world example would provide the human-influenced perturbations on the targets. This type of study would certainly require probabilistic tracking techniques for target initialization and data association.

5.2.3 Additional Image Processing. Having another set of image processing techniques to examine would provide additional dimension to the attribute tracking problem and thus, the integer program. For this reason, the test should use data sets that would require more complex image processing techniques. This would be more beneficial for situations in which the available techniques could provide overlapping data, without any one technique providing a "full picture."

5.2.4 Advanced Data Association Methods. The data association method that was implemented for this thesis was very rudimentary and prone to errors for a variety of reasons. Incorporating more advanced kinematic data association techniques (1) into the tracking algorithm could reduce the potential for the errors and tracking failures that result from this shortcoming. Furthermore, attribute data association could be used as a second check on the validity of the data association that was made. This would require some type of multiple-hypothesis and probabilistic association methodology.

5.2.5 Linear Program Approach. Another area of recommendation involves applying the optimization method in a different way. It is possible that the decision to use the binary integer program formulation was not the best choice. There are other less computationally intensive linear programming methods that can still take advantage of the available information to assign image processing resources.

One potential method is a two-component resource allocation system. The first component of the system would determine how many system resources at a given time can be allocated to targets within the image. Ideally, the problem would only consider how many of each of the several image processing algorithms should be used at a given time, or what time limit should be imposed on the image processing. The second component of the system would be a logic-based system that, given the limitations presented by the first component, determines which of the sub-regions should receive processing. The solution for the first component could be performed using an integer program. Some advantages that may be found with this sort of approach are better adaptivity to changes in target numbers and target types, or adaptability to changes in the system environment that may alter the available resources.

Using a mixed-integer programming (MIP) technique, (12) one could expand the solution space to include a minimum value that each sub-region must have to be considered for image processing. A MIP formulation would be more complex and resource-intensive, jeopardizing any potential gains to be found from utilizing a resource allocation technique. However, the MIP technique is not necessary for this class of problem. A satisfactory level of performance may be found by using some form of Linear Programming or Integer

Programming and solving for the solution based on knowledge of the image space. The benefit of combining different optimization approaches will probably be quickly overcome by the increasing complexity of the resource allocation routine.

5.2.6 Optimized Logic Approach. It is quite possible that, ultimately, the best solution may be found in an approach that combines the speed of the Logic method with the insight of BIP into one quick operation. It is apparent that there are a finite number of ways to allocate resources. Furthermore, a full investigation into the actual assignments made by the resource allocator operating on such a method may find that there are many distinct, common patterns to the solution that depend on only a handful of values. From such information, a library of candidate solutions or solution patterns could be pre-computed and stored for later use. When the conditions that satisfy specified criteria occur, the new Logic algorithm would specify which is the proper resource allocation pattern to implement. Ideally, this method would retain the same insights given by an optimally derived solution, while retaining the ability for quick selection and implementation.

5.2.7 Other Recommendations. More varied values for the gamma-weighting function should be used. A method that would be more proper in terms of maintaining previously acquired information is an exponentially aged weighting filter. This would help stabilize attributes that fluctuate wildly, such as mass. Many of the system parameters used in this thesis were chosen arbitrarily due to a lack of time. It should be possible to determine the optimal solutions for these values through actual empirical analysis. This could lead to improved performance in later implementation.

Bibliography

1. Bar-Shalom, Yaakov, William Dale Blair. *Multitarget-Multisensor Tracking: Applications and Advances*. Norwood, MA: Artech House, Inc., 2000.
2. Bar-Shalom, Yaakov, X. Rong Li Thiagalingam Kirubarajan. *Estimation with Applications to Navigation and Tracking*. New York, NY: John Wiley and Sons, Inc., 2001.
3. Barrick, Sharon, "Linear Correlation." http://www.csus.edu/indiv/b/barricks/EdOctober 2000. Viewed: 21 February 2006.
4. Blackman, Samuel, Robert Popoli. *Design and Analysis of Modern Tracking Systems*. Norwood, MA: Artech House, 1999.
5. Cassandras, Christos G., Liyi Dai Christos G. Panayiotou. "Ordinal Optimization for a Class of Deterministic and Stochastic Discrete Resource Allocation Problems," *IEEE Transactions on Automatic Control*, *43*(7):881–900 (July 1998).
6. Dempster, A. P. "A Generalization of Bayesian Inference," *Journal of the Royal Statistical Society*, *30*(2):205–247 (1968).
7. Linda Shapiro, Azriel Rosenfeld. *Computer Vision and Image Processing*. San Deigo, CA: Academic Press Limited, 1992.
8. Maybeck, Peter S. *Stochastic Estimation and Controls*. New York, NY: Academic Press, 1982.
9. Rafael C. Gonzalez, Richard E. Woods, Steven L. Eddins. *Digital Image Processing Using MATLAB*. Upper Saddle River, NJ: Pearson Prentice Hall, 2004.
10. Sigal, Leonid, Stan Sclaroff Vassilis Athitsos. *Skin Color-Based Video Segmentation under Time-Varying Illumination*. Technical Report, 115 Waterman St. Providence RI 02912: Boston University, April 2003.
11. Suel, Michael, Lawrence O'Gorman Michael J. Sammon. *Practical Algorithms for Image Analysis: Description, Examples, and Code*. New York, NY: Cambridge University Press, 2000.
12. Winston, Wayne L. *Operations Research: Applications and Algorithms* (3rd Edition). Belmont, CA: International Thomas Publishing, 1994.

REPORT DOCUMENTATION PAGE	Form Approved OMB No. 0704-0188

1. REPORT DATE *(DD-MM-YYYY)* 23-03-2006	**2. REPORT TYPE** Master's Thesis	**3. DATES COVERED** *(From - To)* March 2005 - March 2006

4. TITLE AND SUBTITLE Image Processing Resource Allocation Methods for Multi-Target Tracking of Dismounted Targets in Urban Environments	**5a. CONTRACT NUMBER**
	5b. GRANT NUMBER
	5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S) Jon P. Champion	**5d. PROJECT NUMBER**
	5e. TASK NUMBER
	5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Air Force Institute of Technology, Graduate School of Engineering and Management (AFIT/EN), Bldg. 640, 2950 Hobson Way, WPAFB OH 45433-7765	**8. PERFORMING ORGANIZATION REPORT NUMBER** AFIT/GE/ENG/06-13
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) AFRL/SNAT, Attn: Jeffery Layne, Bldg 620 WPAFB, OH, 937-255-1115 x4039 DSN: 785-1115	**10. SPONSOR/MONITOR'S ACRONYM(S)**
	11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED

13. SUPPLEMENTARY NOTES

14. ABSTRACT
Dismounted targets can be tracked in urban environments with video sensors. Real-time systems are unable to process all of the imagery, demanding some method for prioritization of the processing resources. Furthermore, various segmentation algorithms exist within image processing, each algorithm possesses unique capabilities, and each algorithm has an associated computational cost. Additional complexity arises in the prioritization problem when targets become occluded (e.g., by a building) and when the targets are intermixed with other dismounted entities. This added complexity leads to the question "which portions of the scene warrant both low cost and high cost processing?" The approach presented in this thesis is to apply multi-target tracking techniques in conjunction with an integer programming optimization routine to determine optimal allocation of the video processing resources. This architecture results in feedback from the tracking routine to the image processing function which in turn enhances the ability of the tracker.

15. SUBJECT TERMS
Target Tracking; Image Processing; Resource Allocation; Binary Integer Program

16. SECURITY CLASSIFICATION OF:			**17. LIMITATION OF ABSTRACT**	**18. NUMBER OF PAGES**	**19a. NAME OF RESPONSIBLE PERSON**
a. REPORT	**b. ABSTRACT**	**c. THIS PAGE**			LtCol Juan Vasquez
U	U	U	UU	101	**19b. TELEPHONE NUMBER** *(Include area code)* (937) 255-3636, ext 7231

Standard Form 298 (Rev. 8/98)
Prescribed by ANSI Std. Z39.18