Simulating Autonomous Agents in Augmented Reality

Erol Gelenbe [a], Khaled Hussain [b] and Varol Kaptan [a]

[a] *Department of Electrical & Electronic Engineering, Imperial College, London SW7 2AZ*

[b] *School of Computer Science, University of Central Florida, Orlando, Florida 32816*

Abstract

In many critical applications such as airport operations (for capacity planning), military simulations (for tactical training and planning), and medical simulations (for the planning of medical treatment and surgical operations), it is very useful to conduct simulations within physically accurate and visually realistic settings that are represented by real video imaging sequences. Furthermore, it is important that the simulated entities conduct autonomous actions which are realistic and which follow plans of action or intelligent behavior in reaction to current situations. We describe the research we have conducted to incorporate synthetic objects in a visually realistic manner in video sequences representing a real scene. We also discuss how the synthetic objects can be designed to conduct intelligent behavior within an augmented reality setting. The paper discusses both the computer vision aspects that we have addressed and solved, and the issues related to the insertion of intelligent autonomous objects within an augmented reality simulation.

1 Introduction

Discrete event simulation is widely used to model, evaluate and explore operational contexts of real systems under varying synthetic conditions. Simulation runs can predict the capabilities and limitations of a system which is being designed, its ability to operate under different load conditions, or to predict the performance of a system which is either being modified, or to predict the performance of a system which is being evaluated for future operating conditions. Traditionally, discrete event simulation has concentrated on the algorithmic description and control of synthetic entities which are being modeled as they accomplish some meaningful function. Research in simulation has

devoted much attention to appropriate workload representation and output data analysis.

Modern discrete event simulators often use a graphical interface both as an input and as an output medium to simplify and enrich the user's interaction with the simulation both before, during and after the simulation runs. Many simulation tools also provide an animated graphical interface which offers a real-time visual description of a simulation in real time.

A useful and very significant leap forward in simulation technology is to be able to evaluate synthetic simulated conditions in realistic settings. The idea here is to ask questions about "what would happen if ..." in the context of a real environment and actual events. This challenge is the focus of the work addressed in this paper where we mix simulation with reality in real time, in order to examine how novel simulated conditions can actually interact with a real system's operation. This interaction can go in both directions: the course of the real world can be modified by virtual entities, and the virtual objects are constrained to operate in the real world. Mixing reality with simulation in real time raises some very interesting conceptual and practical issues such as:

- How will reality change its behavior as a function of its interaction with synthetic objects, when the real active entities become aware of the behavior of the synthetic entities?
- How will the synthetic entities be programmed to interact with and react to the real (natural) environment?
- How can all these interactions be controlled, programmed and run concurrently in real time?

When one enters into a novel field of research and development to chart unknown terrain, it is both prudent and constructive to base ones work on a practically significant experimental setting, in addition to dealing with the conceptual issues. This is the approach we have taken here. We address some of the issues we have outlined above, and present design principles and solutions in a practical context.

1.1 Augmented Reality for Training Systems

One of the increasingly important application areas of simulation is in education and training, where simulation can be used to illustrate concepts and provide exercises that allow the learner to train in a realistic environment. The use of real scenarios enhanced by "what if" situations offers a very stimulating learning setting for self-learning and self-evaluation.

The use of purely synthetic scenarios in training systems reduces the authen-

ticity of a learning or training exercise, and this can leave the learner with the impression that he/she is interacting with a non-relevant game. The insertion of simulation driven virtual objects in real scenes will offer a higher degree of motivation to the learner, who will face a realistic stimulus approaching that of a real situation under real-time operating constraints. Compared to a real exercise, it will also have significantly reduced costs and hazards. This is particularly true in the field of military training systems where real exercises have costs and hazard levels which approach those of military operations. Important application areas where mixing synthetic and real training environments in a simulation will have major impact on the benefits of the training include:

- Medical education,
- Transportation system management (e.g., control of multiple aircrafts, control of trains, airport management),
- Management of power generation and management systems,
- Financial transactions such as trading in stocks and commodities,
- Management of communication networks.

Our own work is motivated by the design of embedded training, where the term "embedded" means that the training system is built into the actual operational system, and that the operational system and the training system are designed to be used jointly. Many fields of application for augmented reality based training systems have a need for real-time interaction between the learner and the augmented reality which is being observed. Augmented reality will include a significant human sensory environment with a visual component, as well as sound, touch, physical motion and pressure, and even smell. Thus, an augmented reality surgical training operation table, could allow the surgeon to sense the smell of blood and of the chemical products which are being used, as well as to feel the pressure of the organs as the synthetic surgical instruments are being applied to the synthetic patient, whose resulting vital signs and endoscopic images are also being shown on an appropriate set of screens.

In the work we present here we concentrate on the visual component. However many of the principles that we develop in this work are in fact generic, and they can also be used to deal with other media. For instance, we could consider:

- Sound environments mixing real and synthetic sounds,
- A smell environment introducing real and synthetic smells,
- An environment including touch or pressure which are sensed when a human operator manipulates real and synthetic tools,
- Or a multisensory augmented reality environment combining all five senses.

Another important component of this paper is the set of algorithms that we have designed and used to control the motion of the synthetic objects. This part is detailed in Section 3. These algorithms are based on two principles:

- The collection of objects pursue one or more goals, such as attaining a specific destination or a region in the scene. This element introduces our use of Reinforcement Learning as a means for objects to autonomously try to attain these goals [1,2], based on a Random Neural Network implementation [3].
- Additionally, these objects have a collective behavior which is related to two different aspects: (a) in everything they do, they need to take into account patterns of collective behavior – we approach this through Social Potential Fields [4]; (b) furthermore their collective behavior is also influenced by Imitation of the object which may be perceived as the most effective, or of the leader.

1.2 Organization of the Paper

The rest of this paper is organized as follows. Section 2 describes the visual augmented reality system we designed and implemented. A description of our algorithm for injection of synthetic agents in real-world video is presented in section 2.3 and examples of the working system are given in section 2.4.

Section 3 describes our approach to modeling behavior of collections of cooperating agents. The agent model is presented in sections 3.2–3.6, and performance results are discussed in 3.7.

In the Appendix we review some standard image segmentation techniques to provide the reader with background information which can be useful in understanding our approach to identifying objects in natural scenes and registering with the corresponding objects in the synthetic scene.

2 Visual Augmented Reality

A visual augmented reality system creates a combination of a real and virtual scene in which the user perceives a significant difference between the real and augmented world. Figure 1 shows an example of a view that the user might see from an augmented reality system showing a live scene with a virtual tank. We will refer to the real world as a "scene", even though we are in fact dealing with a video sequence representing the world as it is being viewed in real-time. The scene has different visual representations depending on where it is being viewed from, and the viewing pint is often referred to as the "aim-point".

One of the difficult technical issues in augmented reality is the "registration problem", which refers to the need for determining the isomorphism between

Fig. 1. Live scene with virtual tank.

objects and features in a live scene with the corresponding features and the corresponding objects in an augmented version of that scene. Errors in registration will generate visual inconsistencies between real and virtual images with obvious consequences on the value of the augmented reality system for simulation purposes. Many researchers have tried to addressed this issue, and it is generally accepted that registration using only information from a sensor based tracking system cannot achieve a perfect match [5], therefore it is a common practice to improve registration by using of some kind of image processing based algorithm. One approach is to detect features in the real image and uses them to enforce registration, while others have considered placing special marks (e.g. LEDs [5], circles [6], a calibration grid [7]) in the environment. Image-processing algorithms detect the locations of these marks and use them to enforce registration, assuming that one or more special marks are visible at all times. Another approach [6] uses a survey of the live environment with real-time instrumentation, providing more information about objects and their distances in the live environment, but requires specific equipment and significant amounts of additional computation for the interpretation of the sensors' output.

Almost all augmented reality techniques assume that virtual objects and live objects have the same detailed shape. This assumption is only valid for rigid objects such as roads and buildings: many virtual object generators will use a simplified representation, and will even sometimes only make use of templates; e.g., a synthetic pine tree may be some idealized template of a pine tree, rather than the actual pine tree being represented at some location in a scene.

In our work we deal with both rigid and non-rigid objects. Since our system is expected to work in a wide variety of natural settings, we do not make use of special marks in the environment. Thus even if we obtain accurate registration, we may still have visual inconsistencies due to nonrigid objects (e.g., trees). To address this problem, we segment the live image into objects and then register

them with the corresponding virtual objects. We will detail this approach in section 2.3.

2.1 Overview of our Approach

Our approach is postulated on the premise that the real world is far more dense in significant objects, than in the number and type of synthetic objects we insert into it. This is of course totally different than an interactive computer game in which no real objects exist (i.e. everything is artificial), and in which there may be a large number of moving synthetic objects (e.g. aircrafts, robots, etc.). Our system design is based on the following assumptions:

- The 3-D real world of stationary objects, i.e., non-moving such as landmarks, terrain elevation, houses, trees, etc. is represented in a terrain database (TB) which has some desired level of precision.
- There are a limited number of moving real objects and virtual objects, which together cover only a small fraction of the scene being viewed at any time or being represented in the TB.
- Moving virtual objects are also represented in the TB; they are synthetically moved either manually (by the operator or learner), or using a simulation system.
- From the TB, and knowing a specific viewing location and direction (which we call the aim-point), it is possible to synthetically generate a graphics image representing the scene as it is viewed from the TB. This graphics based scene (GS) will include the synthetic objects.
- Inaccuracies in the TB are expected to exist and will often have to be compensated for in real-time.
- The real scene (RS) corresponding to a given aim-point is viewed through an appropriate video camera or sensor.

We have developed a new real-time moving-object injection method based on these assumptions with the purpose of inserting synthetic moving objects into live video in real-time, and involving techniques for image segmentation and registration. Our approach tolerates a certain level of inaccuracy in the TB, and avoids the expense of specific location and range finding instrumentation. The practical outcome of this work is a Target Overlay System (TOS) that will support the Inter-Vehicle Embedded Simulation Technology (INVEST) at the U.S. Army Simulation and Training Command.

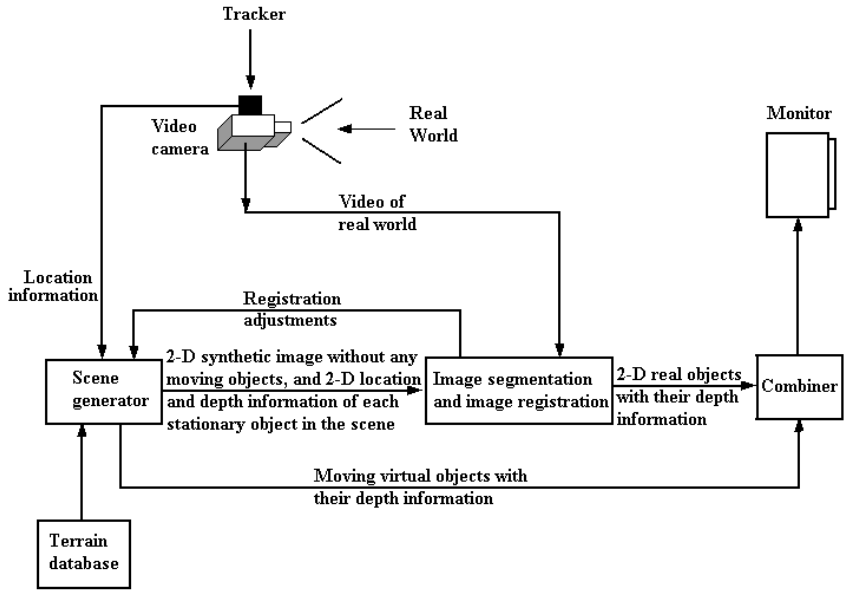


Fig. 2. System architecture.

2.2 System Architecture

A schematic representation of the system we have developed is shown in Figure 2. The first challenge we faced in our design is to determine a representation of the live and virtual environments that could be compared to each other. The simplest method of capture of the **Live Environment** is through a video camera providing color RS-170 video. This provides a two-dimensional representation of the scene. Its main shortcoming is that it provides little information about the precise location and representation of the objects in the scene that could occlude the added virtual objects. Camera pointing angles and camera zoom information are available to a PC on its 1553 data bus. The position and speed of the platform supporting the camera are also provided by an instrumentation system to the 1553 bus. This positional information is used to determine the aim-point so that we may also generate a synthetic image that is equivalent to the live image.

The TB format of terrain we have used is the SAF (Semi-Automated Forces) [8] environment provided via topological terrain formats. Specifically, the Modular Semi-Automated Forces (ModSAF) simulation system uses the Compact Terrain Database (CTDB) format with terrain height and feature representations. It also provides detailed information about any virtual objects which are placed in the virtual TB. Another approach to representing the virtual environment is by visual databases which are used in stealth environment applications. These are often more detailed than topological databases because they were designed to be viewed as realistic renderings by a human user. Visual databases also provide a two-dimensional representation of the virtual environ-

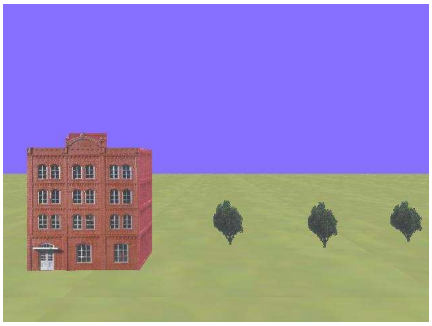
ment, and are much closer to the available live and desired visual augmented representations. For the virtual representation, we have chosen OpenGL which renders three-dimensional virtual scenes in two dimensions. The rendering is often done by hardware, which is much faster than software rendering. As part of the rendering process in OpenGL, the depth (Z-buffer) of each object in the scene is determined. This depth is then used to calculate which objects are visible from the current aim-point.

The block diagram in Figure 2 outlines the flow of data in the system. A camera provides the live terrain image. A tracker is attached to the camera to provide the location and the direction of the camera. The scene generator uses this location information to generate the 2-D synthetic image, and 2-D location and depth information of each stationary object in the field of view. Our image segmentation algorithm uses this information to segment the live image into 2-D real stationary objects. Since we cannot obtain depth information from the live image in real time, we assume that the depth of each 2-D real stationary object is identical to the depth of the corresponding virtual stationary object. The image registration algorithm uses the locations and the sizes of the virtual and real stationary objects to adjust the viewpoint of the virtual scene. The scene generator also generates the moving objects with their depth information. The combiner then inserts these moving objects based on their depth information between real stationary objects.

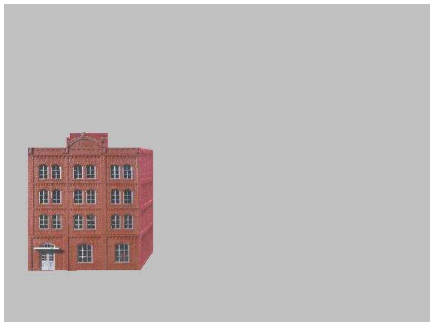
2.3 Moving-Object Injection in Real-Time

At the core of the system we have developed is an algorithm to insert synthetic moving objects into live images in real time. The essence of this approach is that it is purely computationally based, and that it does not make use of any direct video manipulation.

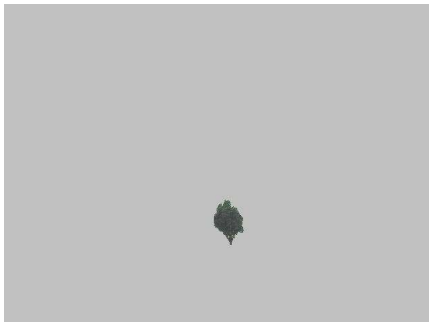
A primary concern is the proper placement of the virtual objects in front of, or behind, live objects. Thus the realistic representation of the inserted objects is tied to both the appropriate occlusions and the shapes and sizes of inserted objects. A good solution to the occlusion problem requires detailed knowledge of the objects and of their location in the live scene. Since the two-dimensional live images provide no prior information about the objects in the scene, we use an image segmentation technique to segment the live image into objects. We then use a registration technique to register objects in the live image with those in the virtual scene. Depth information from the virtual scene is used to associate relative depth information to each object in the live image, so that there is no need for additional instrumentation to calculate the depths of the live world surrounding the observer.



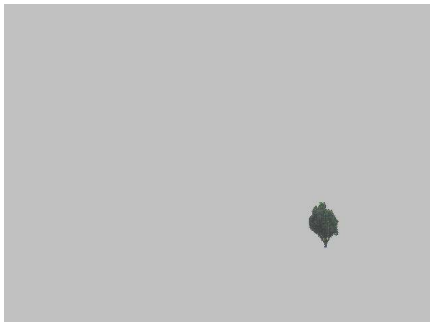
(a) The synthetic image



(b) The synthetic building



(c) The left synthetic tree



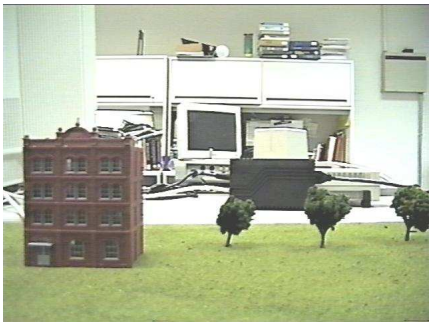
(d) The middle synthetic tree

(e) The right synthetic tree

Fig. 3. The object representation of the virtual scene.

Given objects $(vo_1, vo_2, ..., vo_n)$ of the virtual scene (see Figure 4) with their position and color information, and given a live image, we segment this live image into the same number of objects $(lo_1, lo_2, ..., lo_n)$. Each object lo_i is equivalent to the corresponding virtual object vo_i, but they do not necessarily have the same exact shape, location, and color. Since we cannot obtain depth information from the live image, we assume that the depth of a live object is equal to the depth of the corresponding synthetic object. Representing the virtual live scene by objects provides approximate location and color information for each one of them. This in turn gives the image segmentation algorithm the capability to find the corresponding objects in the live scene in real time.

To segment the live scene into objects we first build a look-up table for each virtual object using its color information with noise. This table is indexed by



(a) Table-top camera scene

(b) The building

(c) The left tree

(d) The middle tree

(e) The right tree

Fig. 4. The decomposition of the live image into objects.

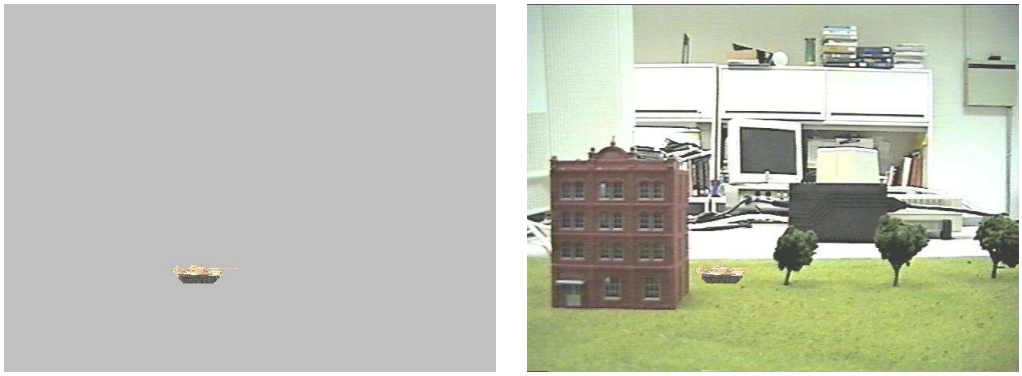
a color vector which allows us to segmentation the real image by applying the look-up tables to each pixel in the image.

2.4 Experimental Implementation

To test our system, we considered several methods to achieve correlated live and virtual images. The first approach was to model terrain so that the virtual terrain model could be adapted to accurately model the real world. The main obstacle with this solution was the generation of both an OpenFlight database and a CTDB database for use with moving synthetic objects such as vehicles. Several OpenFlight databases exist for areas of the University of

(a) (b)

Fig. 5. (a) A synthetic moving-object, and the result (b) of inserting it into the live scene.

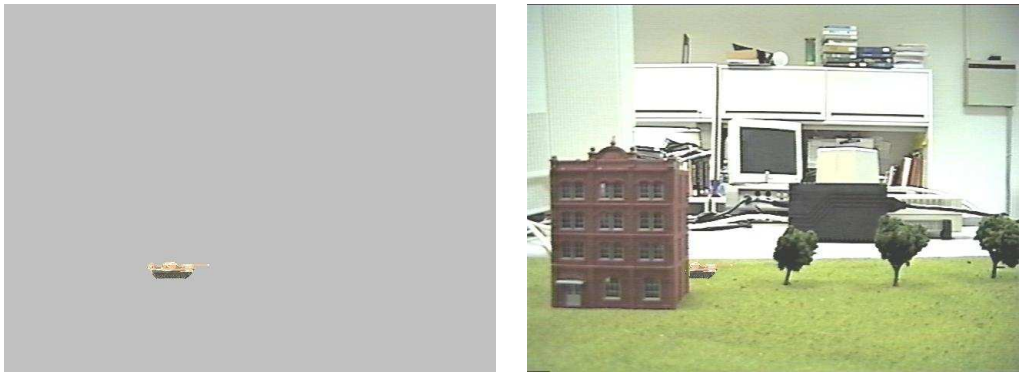


(a) (b)

Fig. 6. (a) A synthetic moving-object, and the result (b) of inserting it into the live scene.

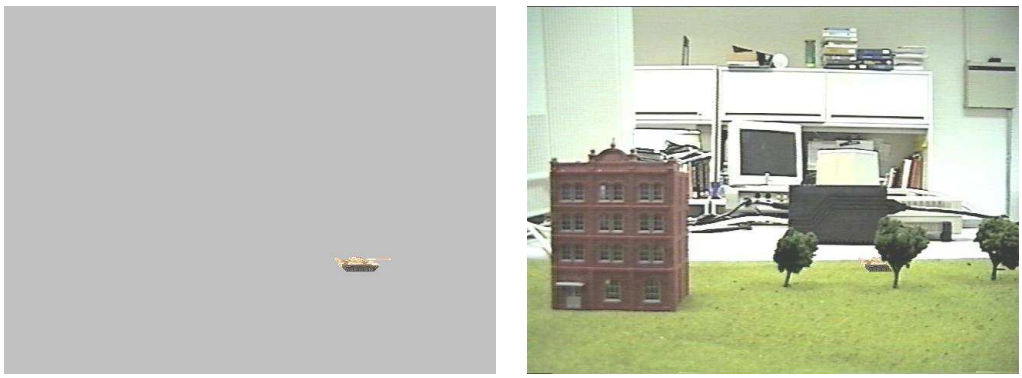
Central Florida (UCF) campus. However, there were no corresponding CTDB databases. Tools are available to convert from OpenFlight to CTDB format, but they require significant manual tweaking of input data. Another obstacle was the ability to isolate aspects of the live terrain. Variable weather would limit the times we could use the live image. The coming and going of cars, bikes, and pedestrians would also change the environment. Even larger structures change with ongoing new building construction at UCF.

We therefore developed a table-top model of the real world to be used for the live camera scene. The table-top represents a geographic area for which we already have correlated OpenFlight and CTDB databases. This approach allows the table-top to be correlated with the virtual terrain. We can also adjust the OpenFlight and CTDB databases to fix any inconsistencies between the real and synthetic views. The table-top also allow us to carry out the development work from "uncontrolled objects" such as unrelated vehicles, pedestrians, animals and adverse weather.



(a) (b)

Fig. 7. (a) A synthetic moving-object, and the result (b) of inserting it into the live scene.



(a) (b)

Fig. 8. (a) A synthetic moving-object, and the result (b) of inserting it into the live scene.

To illustrate the whole experimental approach, an example of the virtual scene is shown in Figure 3, and the corresponding live image and its decomposition into objects are shown in Figure 4. Once the object representation of the live scene is determined, the insertion of a synthetic moving-object will be based on the distance at which this object should be placed. Figure 5, 6, 7, and 8 illustrate the insertion of synthetic-target objects into the live scene.

3 Autonomous Behavior of Injected Objects

The behavior of injected artificial entities can be as important as their appearance in a Visual Simulation. This is especially important in the context of simulations designed for training personnel or evaluating a "what if ..." situation. In such simulations, the behavior of agents will have an important effect on the final outcome in the form of "acquired training experience". Unrealistic agent behavior, e.g., in the form of very limited or even extremely advanced

intelligence, for example during training, will result in poor performance of the trainees in a real-life situation.

Agent behavior in a sophisticated simulated environment can be very complex and may involve many entities. Intelligence can be employed at very different levels. A very simple example will be an agent that has to go from one position to another trying to minimize travel time. A very complex example of intelligent behavior can include the decision to cancel the mission of a group of entities and relocating them as a backup for another group. While the first problem can be easily solved by a single autonomous entity, the second will involve some authority that can make a higher-level decision based on information feedback from the lower-level autonomous agents.

3.1 Related Work

Multi-Agent systems are a very important field in AI since they emerge as a natural way of dealing with problems of distributed nature. Such problems exist in a diversity of areas like military training, games and entertainment industry, management, transportation, information retrieval and many others. The classical approach to AI, until now, has been unable to provide a feasible approach for solving problems of this nature. The need for such tools has lead to the "alternative" approach of behavior-based systems, popularized by the works of Brooks [9] and Arkin [10]. This approach takes simple behavior patterns as basic building blocks and tries to implement and understand intelligent behavior through the construction of artificial life systems. Its inspiration comes from the way intelligent behavior emerges in natural systems studied by Biology and Sociology. Good discussions on the development of behavior-based AI can be found in [11,12] and an extensive treatment of the subject is given in [13].

Multi-agent systems interacting with the real world face some fundamental restrictions. Some of these are:

(1) They have to deal with an unknown and dynamic environment
(2) Their environment is inherently very complex
(3) They have to act within the time frame of the real world
(4) The level of their performance should be "acceptable"

In order to meet these requirements, agents have to be able to learn, coordinate and collaborate with each other. Reinforcement Learning emerges as the "natural way" of dealing with the dynamism and uncertainty of the environment. The complexity of the environment and the strict timing constraints however make the learning task extremely difficult. Even simple multi-agent systems consisting of only a few agents within a trivial environment can have

prohibitively expensive computational requirements related to learning [14–16].

The behavior-based systems have been more successful at dealing with such problems. Biology-inspired models of group behavior such as Reynold's boids [17] and approaches based on potential fields [4] are able to address group behavior at a reasonable cost. Because of their performance and ability to scale better, they have been widely employed in technology-driven fields such as the computer-games industry [18,19].

One of the main problems of behavior-based systems is that their constituents can be very easily caught in local-minima. The question of how to combine different (possibly conflicting) behaviors in order to achieve an emergent intelligence is also very difficult. Multi-Agent Reinforcement Learning in the behavior domain [20,21] is an actively explored approach to solve these problems in a robust way.

3.2 The Agent Model

The design of the agent model is based on the assumption that agents will perform "outdoor" missions in a terrain which is relatively sparse with respect to obstacles and enemies. It is not very suitable for "indoor" missions like moving inside a building or a labyrinth, where a more specialized approach will be required. A "mission" in our model is defined as the problem of going from some position A to some other position B avoiding being hit by an enemy or crashing into the natural and artificial obstacles present in the terrain. The success of the mission is measured by the amount of time necessary for the whole group to complete the goal and the survival rate.

Our approach is based on a hierarchical modular representation of agent behavior. This method allows for de-coupling the task of group navigation into simpler self-contained sub-problems which are easier to implement in a system having computational constraints due to interaction with real-life entities.

Different decision mechanisms are used to model different aspects of the agent behavior and a higher level coordination module is combining their output. Such an architecture allows "versatile agent personalities" both in terms of heterogeneity (agent specialization) within a group and dynamic (i.e. mission-context sensitive) agent behavior.

The hierarchical modularity of the system also facilitates the assessment of the performance of separate components and related behavior patterns on the overall success of the mission.

In our current model, we have three basic modules that we call the *navigation* module, *grouping* module and *imitation* module.

- The Navigation Module is responsible for leading a single agent from a source location to a destination location, avoiding danger and obstacles.
- The Grouping Module is responsible for keeping a group of agents together in particular formations throughout the mission.
- The Imitation Module is modeling the case when an inexperienced agent will try to mimic the behavior of the most successful agents in the group and thus increase its chances of success.

The decisions of these modules are combined at a higher-level module called the *Coordinator Module*.

3.3 Coordination of Behavior Modules

The current model of behavior combination is to get, at each time step, a weighted sum of the separate decisions recommended by each basic module, where the decisions are in the form of a 2D vector representing a request to move in a particular direction with a particular speed:

$$\vec{V}_{\text{overall}} = k_{\text{nav}} * \vec{V}_{\text{nav}} + k_{\text{grp}} * \vec{V}_{\text{grp}} + k_{\text{imt}} * \vec{V}_{\text{imt}}$$

The coordinator can for example give priority to the Navigation Module and inhibit the others when it detects that they cause an agent to be trapped in a local minimum. The leader of a group will also honor the Navigation Module, expecting group members to follow him. Another example is when emphasis is given to the Grouping Module, helping a wounded or important agent to stay close to the other members so that it is well protected.

Another degree of freedom comes from the ability of the coordinator to see the "bigger picture" and not only judge how much a module should affect the final outcome, but also give a constructive feedback on how a module should adjust its internal parameters for the good of the mission. The basic decision modules that we consider in this work are described in the following subsections.

3.4 Navigation Module

For the purpose of simplicity and efficiency, the Navigational Module generates moves based on a quantized representation of the simulated environment in the form of a grid. Each cell in the grid represents a position and an "agent

action" is defined as the decision to move from a grid cell to one of the eight neighboring cells. A succession of such actions will result of a completion of a mission. The agents can also access terrain-specific information about features and obstacles of natural (trees, etc.), and artificial origin (buildings, roads, etc.) and also presence of other (possibly hostile) agents. The interaction between an enemy (a hostile agent) and an agent is modeled by an associated risk. This risk is expressed as a probability of being shot (for an agent) at a position, if the position is in the firing range of an enemy. The goal of the agent is to minimize a function G (which in this case is the estimated time of a safe transit to the destination). We use G to define the Reinforcement Learning Reward function as $R \propto 1/G$.

Successive measured values of R are denoted by R_l, $l = 1, 2, \ldots$. These values are used to keep track of a smoothed reward

$$T_l = bT_{l-1} + (1 - b)R_l, \quad 0 < b < 1$$

where b is close to 1. A Navigational Module of an agent has a so-called "cognitive map" which is a collection of latest and smoothed rewards for each decision taken at each visited grid cell.

The decision-making element of a Navigation Module is a fully-connected RNN network consisting of 8 neurons (each representing a possible decision). The training is performed by reinforcing the weights of each neuron, depending on the difference between the latest and smoothed rewards; positive difference indicates improvement and negative difference indicates deterioration. A detailed description of the network and the learning process are presented in [2].

By using previously acquired information and current sensory input, an agent can start with near-optimal estimates of the rewards and skip an otherwise prohibitively-long learning session and focus on adapting to the dynamic changes in the environment.

3.5 Grouping Module

Grouping behavior module is based on the idea of social potential fields [4] which is a simple distributed-control approach inspired by the attractive and repulsive forces between charged particle in physics. Although this method has been used in a broader domain (including path-planning), we restrict its usage only to model grouping behavior for which it is particularly well suited. Using potential fields methods for other purposes like generalized navigation and obstacle avoidance requires dealing with local minima problems and difficult

to design force-configurations that can easily nullify the simplicity gained by using the method in the first place.

In our treatment, we restrict the form of the force between agents i and j to:

$$\vec{V}_{i,j} = \left(-\frac{a}{r^{\alpha}} + \frac{b}{r^{\beta}} \right) \hat{r}$$

where a, b, α, β are dynamic parameters and the force vector $\vec{V_{i,j}}$ describes the effect of the position of agent j on the decision of agent i. When there is a stable equilibrium point, an entity experiencing such a force will stay at a distance R_0 from the force source, where

$$R_0 = \left(\frac{b}{a} \right)^{\alpha - \beta}$$

The total effect on agent i can be calculated as:

$$\vec{V}_{\mathrm{grp}_i} = c * \sum_j \vec{V}_{i,j}$$

By varying the parameters of each force, different behaviors like attraction to an agent, repulsion from an agent or trying to stay within some distance from an agent can be modeled - - the last being especially important in forming spatially localized groups.

These behaviors are very similar and can be used to get the effect of the *collision avoidance* and *flock centering* rules as described by Reynolds [17].

By setting up a two-way mesh of forces between a number of agents, for example, a spatially localized group can be created that will try to stay together.

Another simple example is a one-way mesh of forces from the leader of the group to the other members, suggesting that they should stay close and follow if necessary the leader, without having any effect on his decision making.

3.6 Imitation Module

The imitation module proposes a decision which is a weighted sum of the navigational decisions of some of the members of the agent group:

$$\vec{V}_{\mathrm{imt}_i} = \sum_{j \in S} w_j * \vec{V}_{\mathrm{nav}_j}$$

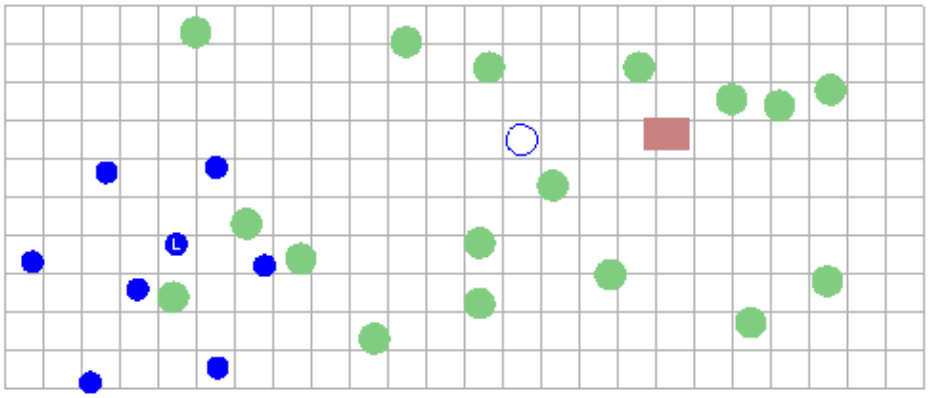


Fig. 9. Initial Agent Configuration

The weight distribution can be dynamic, in order to reflect the group members which are currently observable or known to be experienced, for example. The purpose of imitation is to efficiently take advantage of experience without going through the trouble of actually acquiring it - that is, it has a much lower computational cost, compared to the other methods.

The *velocity matching* flock behavior described in the work of Reynolds [17] which he defines as "attempt to match velocity with nearby flocks" is a very similar idea.

3.7 Experimental Results

We ran a simple navigation mission on our table-top database in order to show the effects of the different behavior modules on overall performance. The database and the initial agent configuration is shown in Figure 9. There are 8 agents (1 leader and 7 group members) represented by blue(dark) filled circles and their mission is to go to the final destination (hollow circle). The leader agent is marked with the letter 'L'. There are trees and a building in the terrain represented by green(light) filled circles and a red(light) filled rectangle. The grid represents the quantization of the terrain.

There are two types of SPF forces involved: (a) A two-way force (F_1) between group members (b) A one-way force (F_2) from the leader to the group members. The force parameters are: $F_1(a = 1, \alpha = 1.6, b = 16, \beta = 3.6)$ and $F_2(a = 1, \alpha = 1.6, b = 4, \beta = 3.6)$. The parameters have the effect of keeping an inter-group distance of approximately 4 units, and distance between group members and the leader of approximately 2 units - in this way, the group is surrounding the leader.

A simple metric defining the quality of a spatial agent configuration (with respect to SPF) is to sum the magnitudes of the total forces exerted on each

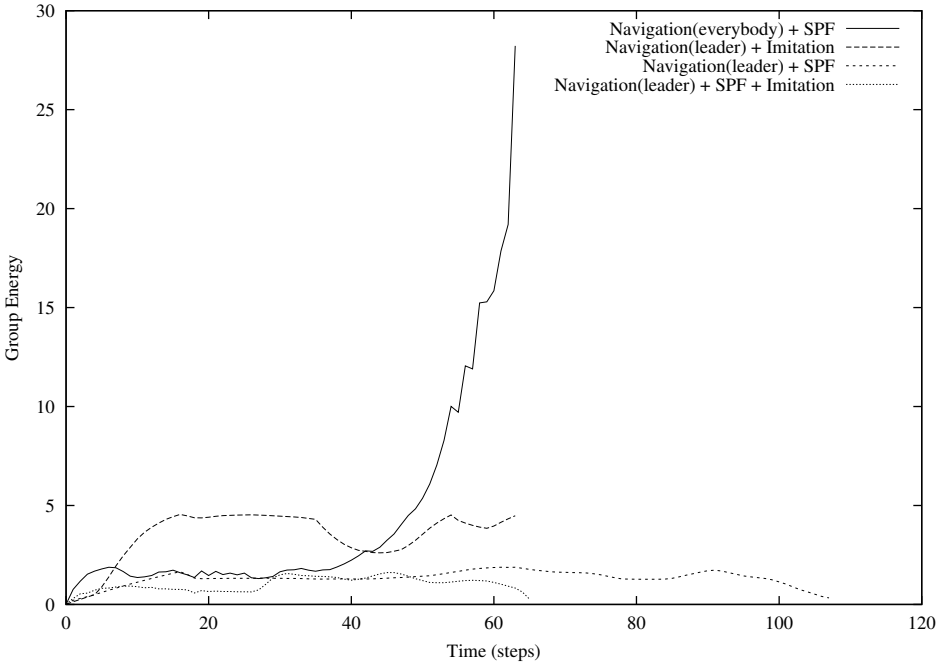


Fig. 10. Group Performance Results

agent. We call this value 'group energy' - a group that is at equilibrium will have this value minimal. The 'group energy' as a function of simulation steps for these cases is plotted in Figure 10.

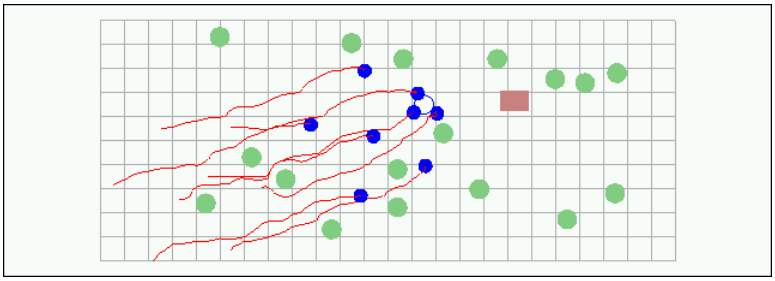
Figure 11 shows the behavior of the agents under different configurations:

(a) All the agents use Navigation and SPF.
(b) The leader uses Navigation, others use Imitation.
(c) The leader uses Navigation, others use SPF.
(d) The leader uses Navigation, others use SPF and Imitation.

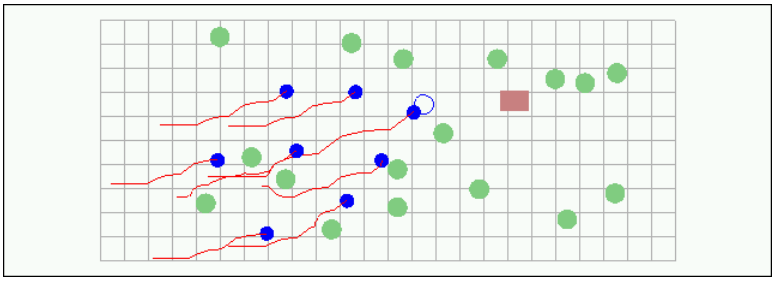
The performance of (a) is good until agents approach the destination and pack together trying to occupy the destination grid cell (The packing could have been avoided by assigning each agent a different non-overlapping destination or monitoring progress and turning off Navigation for group members once the goal is close enough). This is the computationally most expensive run, since every agent has its Navigation Module running.

The performance of (b) is worst overall, since the SPF module is disabled during this run. The Imitation module still manages to somehow keep the group together, although not very successfully.

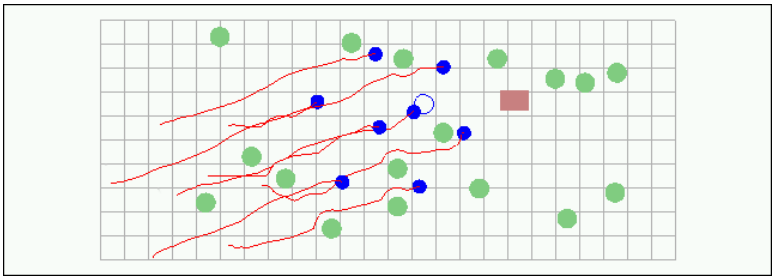
The performance of (c) is very good, but it takes longer for the group members to settle down once the leader reaches the goal. This is because group members are 'dragged' by the leader and follow him from behind.



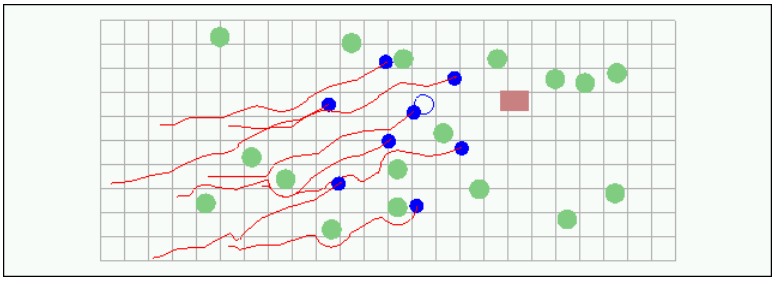
(a) Navigation(everybody) + SPF



(b) Navigation(leader) + Imitation



(c) Navigation(leader) + SPF



(d) Navigation(leader) + SPF + Imitation

Fig. 11. Routes taken by agents in different behavior configurations.

The performance of (d) is the best, the group reaches the destination quickly preserving the group formation. When the leader reaches the destination, the group members are well placed around him. It is interesting to note that SPF + Imitation, maintains on the average a smaller 'group energy' than SPF alone, most probably because of the predictive nature of the Imitation module.

4 Conclusions

Modern discrete event simulators often require the representation of complex autonomous behaviors within a visually realistic setting. They often use a graphical interface both as an input and as an output medium to simplify and enrich the user's interaction with the simulation both before, during and after the simulation runs. Many simulation tools also provide an animated graphical interface which offers a real-time visual description of a simulation in real time.

A useful and very significant leap forward in simulation technology is to be able to evaluate synthetic simulated conditions in realistic settings. The idea here is to ask questions about "what would happen if ..." in the context of a real environment and actual events. This challenge is the focus of the work addressed in this paper where we mix simulation with reality in real time, in order to examine how novel simulated conditions can actually interact with a real system's operation. This interaction can go in both directions: the course of the real world can be modified by virtual entities, and the virtual objects are constrained to operate in the real world.

In this paper we have discussed the conceptual issues which arise in this key area of simulation, and we have presented some design principles and a practical implementation. Key issues we covered in this paper include a new method for injecting moving synthetic objects in real-time into real world video based on terrain databases, graphics rendering and image segmentation, and a novel approach to automatically control the motion of synthetic agents within the realistic live scene. The experiments show that our modular behavior-based approach is able to combine simple behavior modules such that the emergent composite behavior outperforms each of its constituents.

A Review of Image Segmentation Techniques

Image segmentation is a partitioning of an image into a set of elementary regions characterized by the fact that some property is constant in each elementary region. Most image segmentation approaches belong to one of the following classes:

(1) **Segmentation by region growing techniques**
The main idea of the segmentation by region growing techniques [22–24] can be summarized in the following steps:
(a) Label each image pixel as a separate region.
(b) Calculate certain criterion value (e.g., color variance) between each

spatially adjacent regions.
   (c) Merge all pairs of spatially adjacent regions that meet this criterion value.
   (d) If there are no two regions can be merged, stop.

(2) **Segmentation by split and merge techniques**

The basic idea behind the segmentation by split and merge techniques [25,26] can be described using the following steps:
   (a) Label each block in the image as a separate region.
   (b) If any region doesn't satisfy certain criterion, split it into four regions.
   (c) If any two adjacent regions satisfy certain criterion, merge them into one region.
   (d) If no region can be split or merged, stop.

(3) **Texture Based segmentation techniques**

Texture based segmentation techniques can be classified into the following categories:
   (a) Model based texture segmentation [27,28].
   (b) Textured segmentation using artificial neural network [29,30].
   (c) Statistical texture segmentation [31,32].
   (d) Texture segmentation using filters [33,34].

(4) **Motion based segmentation techniques**

Motion segmentation refers to grouping pixels that have common motion. Motion segmentation methods [35,36] are useful for representing a video shot as a background object plus moving foreground objects. This produces an MPEG-4 representation for video coding. For example, if a scene contains only a speaker and a background as in a video-conference sequence, the speaker (without the background) is considered as an object and the background is considered as another object. The benefit of this representation is that it allows each object to be compressed separately and with a different algorithm which can achieve the best quality/compression trade-off.

(5) **Pixel based segmentation techniques**

In pixel image segmentation, color features of pixels are used to divide an image into a set of elementary regions. Pixel image segmentation techniques can be broadly divided into the following groups:
   (a) Histogram based techniques [37–39].
   (b) Segmentation by clustering data in color space [40,41].

References

[1] R. S. Sutton, Learning to predict by the methods of temporal differences, Machine Learning 3 (1988) 9–44.

[2] E. Gelenbe, E. Şeref, Z. Xu, Simulation with learning agents, Proceedings of IEEE 89 (2) (2001) 148–157.

[3] E. Gelenbe, Learning in the recurrent random neural network, Neural Computation 5 (1) (1993) 154–164.

[4] J. H. Reif, H. Wang, Social potential fields: A distributed behavioral control for autonomous robots, in: A. K. Peters (Ed.), International Workshop on Algorithmic Foundations of Robotics (WAFR), Wellesley, Massachusetts, 1995, pp. 431–459.

[5] M. Bajura, U. Neumann, Dynamic registration correction in video-based augmented reality systems, IEEE Computer Graphics and Applications 15 (5) (1995) 52–60.

[6] J. P. Mellor, Enhanced reality visualization in a surgical environment, Master's thesis, MIT, Department of Electrical Engineering and Computer Science (Jan. 1995).

[7] S. W. Lawson, J. R. Pretlove, Augmented reality for underground pipe inspection and maintenance, in: International Symposium on Intelligent Systems and Advanced Manufacturing, Telemanipulator and Telepresence Technologies V, Vol. 3524 of Proceedings of SPIE, Boston, USA, 1998, pp. 98–104.

[8] E. Gelenbe, Modeling CGF with learning stochastic finite-state machines, in: Eighth Conference on Computer Generated Forces and Behavioral Representation, Orlando, Florida, 1999, pp. 113–115.

[9] R. A. Brooks, A robust layered control system for a mobile robot, IEEE Journal of Robotics and Automation RA-2 (1) (1986) 14–23.

[10] R. C. Arkin, Motor schema-based mobile robot navigation, International Journal of Robotics Research 8 (4) (1989) 92–112.

[11] L. Steels, The artificial life roots of artificial intelligence, Artificial Life 1 (1993) 75–110.

[12] R. A. Brooks, Cambrian Intelligence: The Early History of The New AI, The MIT Press, Cambridge, Massachusetts, 1999.

[13] R. C. Arkin, Behavior-Based Robotics, The MIT Press, Cambridge, Massachusetts, 1998.

[14] M. Tan, Multi-agent reinforcement learning: Independent versus cooperative agents, in: Tenth International Conference on Machine Learning, Amherst, Massachusetts, 1993, pp. 330–337.

[15] N. Ono, K. Fukumoto, Multi-agent reinforcement learning: A modular approach, in: Proc. of Second International Conference on Multi-Agent Systems, AAAI Press, Kyoto, Japan, 1996, pp. 252–258.

[16] N. Ono, K. Fukumoto, A modular approach to multi-agent reinforcement learning, in: G. Weiss (Ed.), Distributed Artificial Intelligence Meets Machine Learning, Vol. 1221 of Lecture Notes in Artificial Intelligence, Springer-Verlag, 1997, pp. 25–39.

[17] C. W. Reynolds, Flocks, herds, and schools: A distributed behavioral model, Computer Graphics 21 (4) (1987) 25–34.

[18] C. W. Reynolds, Steering behaviors for autonomous characters, in: Game Developer Conference, San Jose, California, 1999, pp. 763–782.

[19] D. C. Pottinger, Implementing coordinated movement, Game Developer Magazine (1999) 48–58.

[20] M. J. Mataric, Reinforcement learning in the multi-robot domain, Autonomous Robots 4 (1) (1997) 73–83.

[21] T. Balch, Behavioral diversity in learning robot teams, Ph.D. thesis, Georgia Institute of Technology (1998).

[22] C. R. Brice, C. L. Fennema, Scene analysis using regions, Artificial Intelligence 1 (3-4) (1970) 205–226.

[23] J. A. Feldman, Y. Yakimovsky, Decision theory and artificial intelligence: I. A semantics based region analyzer, Artificial Intelligence 5 (4) (1974) 349–371.

[24] O. Monga, An optimal region growing algorithm for image segmentation, International Journal of Pattern Recognition and Artificial Intelligence 1 (4) (1987) 351–375.

[25] F. Cheevasuvit, H. Maitre, D. Vidal-Madjar, A robust method for picture segmentation based on a split-and-merge procedure, Computer Vision, Graphics, and Image Processing 34 (3) (1986) 268–281.

[26] S. L. Horowitz, T. Pavlidis, Picture segmentation by a directed split and merge procedure, in: Second International Joint Conference on Pattern Recognition, Copenhagen, 1974, pp. 424–433.

[27] B. S. Manjunath, R. Chellappa, Unsupervised texture segmentation using markov random field models, IEEE Transactions on Pattern Analysis and Machine Intelligence 13 (5) (1991) 478–482.

[28] C. S. Won, H. Derin, Unsupervised segmentation of noisy and textured images using markov random fields, Graphical Models and Image Processing 54 (4) (1992) 308–328.

[29] L. Hepplewhite, T. J. Stonham, Unsupervised texture segmentation by hebbian learnt cortical cells, in: 13th International Conference on Pattern Recognition, Vol. D, 1996, pp. 381–385.

[30] B. Tian, M. R. Azimi-Sadjadi, T. H. V. Haar, D. Reinke, Neural network-based cloud classification on satellite imagery using textural features, in: International Conference on Image Processing, Vol. 3, 1997, pp. 209–212.

[31] V. Murino, C. Ottonello, S. Pagnan, Noisy texture classification: A higher order statistics approach, Pattern Recognition 31 (4) (1998) 383–393.

[32] B. S. Runnacles, M. Nixon, Texture extraction and segmentation via statistical geometric features, in: International Conference on Image Processing, Vol. C, Lausanne, Switzerland, 1996, pp. 129–132.

[33] M. Clark., A. C. Bovik, Experiments in segmenting texton patterns using localized spatial filters, Pattern Recognition 22 (6) (1989) 707–717.

[34] T. Randen, J. H. Husøy, Novel approaches to multichannel filtering for image texture segmentation, in: B. G. Haskell, H.-M. Hang (Eds.), Visual Communication and Image Processing, Vol. 2094 of Proceedings of SPIE, 1993, pp. 626–636.

[35] A. M. Tekalp, Digital Video Processing, Prentice-Hall, Upper Saddle River, New Jersey, 1995.

[36] J. Y. A. Wang, E. H. Adelson, Representing moving images with layers, The IEEE Transactions on Image Processing Special Issue: Image Sequence Compression 3 (5) (1994) 625–638.

[37] X. Lin, S. Chen, Color image segmentation using modified HSI system for road following, in: IEEE International Conference on Robotics and Automation, Vol. 3, Sacramento, California, 1991, pp. 1998–2003.

[38] Y. Ohta, T. Kanade, T. Sakai, Color information for region segmentation, Computer Graphics and Image Processing 13 (3) (1980) 222–241.

[39] S. Tominaga, A color classification method for color images using a uniform color space, in: Tenth International Conference on Pattern Recognition, Vol. 1, Atlantic City, New Jersey, 1990, pp. 803–807.

[40] M. Celenk, A recursive clustering technique for color picture segmentation, in: International Conference on Computer Vision and Pattern Recognition, Ann Arbor, MI, 1988, pp. 437–444.

[41] S. E. Umbaugh, R. H. Moss, W. V. Stoecker, G. A. Hance, Automatic color segmentation algorithms with application to skin tumor feature identification, IEEE Engineering in Medicine and Biology Magazine 12 (3) (1993) 75–82.