**Figure 1** A .

The increase throughput

|  |  |  |
| --- | --- | --- |
| Macintosh HD:Users:toni:MindsEye:New:live-bg2:Picture 1.png | Macintosh HD:Users:toni:MindsEye:New:live-bg2:Picture 2.png | Macintosh HD:Users:toni:MindsEye:New:live-bg2:Picture 3.png |
| Macintosh HD:Users:toni:MindsEye:New:live-bg2:Picture 4.png | Macintosh HD:Users:toni:MindsEye:New:live-bg2:Picture 5.png | Macintosh HD:Users:toni:MindsEye:New:live-bg2:Picture 7.png |
| Macintosh HD:Users:toni:MindsEye:New:live-bg2:Picture 8.png | Macintosh HD:Users:toni:MindsEye:New:live-bg2:Picture 9.png | Macintosh HD:Users:toni:MindsEye:New:live-bg2:Picture 10.png |
|  |  |  |

Fig. Frames illustrate a action/activity? Of passing an object from one person to another person. Person 1 was in state ‘walk and carry object in right hand’, transition to ‘stop and carry object in right hand’, transition ‘stop and hands object in right hand” to stop and no object’ to ‘stop and receives object in left hand; to walk and object in left hand.



1. b) c)

Fig. Handing of an object from Person 1 (left) to Person 2 (right) in different real (a,b) and synthetic environments c)

Amit

Paul’s summary:

Uses a classification scheme to analyze video. Once objects are detected, finds velocities and optical flow histograms, then maps these into classifications of motions.

Learns motion models from long video examples, representing objects as nodes and attributes (including motions) as weighted links. Learns links and weights, thus learning new patterns of activity.

Hierarchically matches observations from video into classification scheme.

Preliminary results exist for a small set of human activities, showing some invariance under change of view.

Figure 1 is good and works well to explain feature extraction.

Tables 1, 2, 3 could be used to show that preliminary results exist.

This work is in the extraction module.

**Task 1: Complex activity recognition from tracks:**

The dynamical interactions between objects in a scene can be described using the following characterization: kinesics of individual objects (e.g., walking, running), chronemics or temporal aspects (e.g., standing in a line)), proximics or spatial relationship between objects (e.g., approaching), and haptics, (e.g,. shaking hands, exchanging) (*Anderson, Peter A., Nonverbal Communication: Forms and Functions. 2nd Ed, Waveland Press, Inc., Long Grove, IL, 2008)*. Most work in activity recognition has concentrated on analyzing only one of these aspects (predominantly kinesics). By complex activities, we mean ones that require at least two labels, possibly from the same or different classes. It can range from a person stooping to pick up something while walking (least complex) to understanding the interactions of groups of people and other objects like cars and buildings (highly complex). Our goal of understanding complex activities from video would be to categorize them based on this classification scheme.

Our scheme is hierarchical in the sense that it will be able to represent the activities of a single object, as well as the interrelationships between objects. We will start with the trajectories of the detected objects. By analyzing a small spatio-temporal volume around each detection, we will compute some statistics of that region. For low-resolution scenes or rigid objects, where only the overall motion of the center of mass of the object can be inferred, these could the statistics of the change in the direction of motion. These will be called Histograms of Temporal Angular Gradients (HTAGs). For higher resolution objects with articulated motion (e.g., people), it could also consist of histograms of optical flows (HOFs) describing the motion of the individual parts of the object, thus allowing us to differentiate, for example, between a person standing and a person standing and waving. We will also compute the histogram of relative distance (HRDs) between the detected objects. Thus, for N objects over a small time window (in preliminary implementations we have used about 3 seconds of data), we have N HTAGs and HOFs and (NC2) HRDs. The combination of these histograms over this small time window is called a “motion-word”. (Note that this is a more robust definition that what we have currently as explained in Part C). Over time, each of these histograms are linked together like a string leading to strings of motion-words (SoMs). See Fig. 1 for a simple version of this. The number of motion words can be different in the time windows, but the temporal relationship between the words will be based on the trajectories. They represent the individual motion of the objects as well as their pairwise interrelationships.

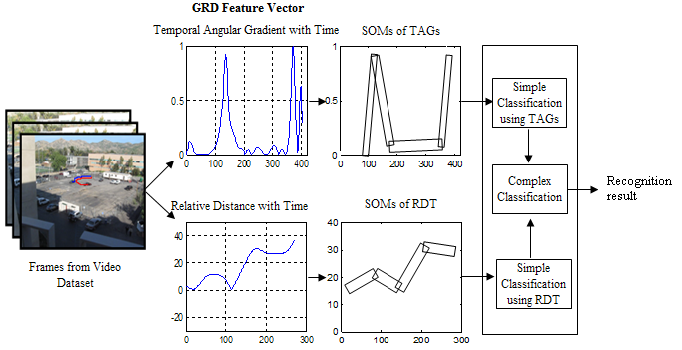
**Task 2: Learning activity-specific context models**

Given observations of activities in an area over extended time periods, we will learn models that describe the dynamical patterns of activities in that area. We shall call this as learning activity patterns or normalcy models. The detected objects will be represented as nodes in a graph with certain attributes, X(t). Some attributes, like the position of the targets or their actions, may be observed, while others may be left to the learning algorithms to infer. Links between the nodes will have weights, W(t), which may be dependent on factors like the distance between the nodes and their relative motion. The weights and attributes will change over time. Thus the interaction between multiple objects can now be represented as a continuous-time stochastic process Z(t) = (X(t), Y(t)). Given observations of detected objects in video, the problem will be to learn the parameters of this model. In principle, this is similar to problems in social network studies for understanding the relationships between various people in the network. Expectation maximization algorithms have been proposed as an efficient solution to this problem [22] because they can work with only partially observed data. Our goal will be to leverage research in estimating the parameters of such continuous time stochastic models to learn normalcy models in video over long observation periods.

**Task 3: Context Aware Activity Recognition**

The final goal will be to use the activity models described in Task 1 with the context models learned in Task 2 to search a database for activities that match a query. Query patterns that do not fit into the learned activity or context models can then be identified as anomalous activities. We propose to do a hierarchical search for this purpose, so that the computationally intensive part is done only on a small portion of the database. First we will compute the probability of the query to belong to a learned context model. We will design probabilistic distance measures for this purpose. This can be done a priori as an indexing scheme. A clip that is far from all the learned context models can be identified as abnormal. Based on this, we will identify certain regions of the database that are more likely to contain clips similar to the query. Then we do a detailed search of these regions by matching the SoM features in Task 1.

Matching of SoMs will be based on comparing strings from the query to the ones from the database. A simple mechanism for matching the activities of Q query objects against D >= Q database objects is to consider every (DCQ) combination of SoMs. Since the SoMs are time sequences, they can be matched using dynamic time warping. We will investigate more efficient matching schemes based on dynamic graph matching techniques to overcome the complexity of the combinatorial search. The relative distance computation between the HTAGs/HOFs naturally lends itself to a dynamic graph representation.

****

**Figure1**: A simple diagrammatic representation of the activity modeling and query classification approach using SoMs. The “string of motion-words” is shown in black using double lines. For simplicity, the SoMs are shown only for the angular gradient and relative distance and termed as GRD (gradient + relative distance).

**Main Contributions**

We believe that one of the main innovations is in describing a feature vector that is able to represent the complex interactions between multiple objects as a function of time. It combines track and non-track based descriptors by leveraging the strengths of video-words and trajectories. The SoM is generalizable to large numbers of objects. By embedding the interrelationships within a graphical structure, we have an elegant representation of changing positions of the various objects as a function of time.

Another major contribution is in leveraging a rich body of work in dynamic social networks to learn activity-specific context models from video. To the best of our knowledge, there has been little work in this direction. Some ad-hoc methods have been proposed for learning dynamic contexts, but it is largely an unexplored area. It has the potential for a large payoff by providing a principled way to take into account context in dynamic environments.

The hierarchical search by considering context into the similarity computation scheme is related to work in object search that combine top-down and bottom-up strategies. No such process exists for recognizing activities. It also has parallels to the neurobiological model of perception that combines gist and saliency for recognition. We are proposing computational processes that parallel this mechanism. We have recently taken some preliminary steps in this direction [A].

[A] RemovedText.docx  
R. Sethi, A. Roy-Chowdhury, S. Ali, Intl. Conf. on Pattern Recognition, 2010

In summary, the main new contributions would be the following:

1. Methods for representing complex interactions among different entities in a video.

2. Learning activity specific models of context in dynamic scenarios by leveraging methods in dynamic social network analysis.

3. Robust activity search by integrating context and saliency models.

**Preliminary Results**

We provide here a short background on some of our recent work on which the proposed tasks are built.

**Feature Vectors:**

We proposed an elementary Strings of Motion-words (SOMs) based representations of activities for simple and complex activity recognition [B]. Each individual track is represented by its gradients as a function of time. Each pair of tracks is represented by the relative distance between the components as a function of time. We shall call this the GRD (gradient + relative distance) feature vector. Please note that the proposed task has a more robust definition of SoMs, a more efficient matching methodology, and builds more complex models for learning and recognition.

[B] [Query-based Retrieval of Complex Activities using “Strings of Motion-Words”](http://www.ee.ucr.edu/~amitrc/wmvc09-2.pdf), U. Gaur, B. Song, A. Roy-Chowdhury, IEEE Workshop on Motion and Video Computing, 2009.

Our track based activity recognition approach using SOMs has the following parts:

1. Extraction of low-level features (GRD) from the tracks. The features will be specific to a single track (gradients), as well as encode the relationships between pairs of tracks (relative distances).
2. In this feature space, we will cluster the features temporally, where each cluster encodes some characteristic of the motion of the objects. These clusters can be thought of as "motion-words", and the entire track as a "string of motion-words". Note that the temporal ordering of these words is critical and the clustering allows us to deal with the noisy data.
3. Then, we will match the "strings" using dynamic time warping to build classification.

Some results on using this methods are shown in Tables 1 and 2.

Table 1 and 2 show some video search results of our track based activity recognition approach using SOMs representation.



**Table 1**: Precision/Recall values for various pre-defined activities as queried in the combined dataset (with object detection).



**Table2**: Precision/Recall values for various pre-defined activities as queried in the combined dataset (without object detection).

**Activity Matching Using Multiple Criteria**

We proposed computational equivalents of neurobiological mechanisms to integrate motion and form/shape to do activity recognition [A]. It resulted in a hierarchical search method whereby one of the features was used to first cluster the database, followed by a detailed search using the other feature. The method relied on multiple hypothesis testing using statistical bootstrapping methods to find the confidence intervals for each feature vector. The distance measurements using our method on KTH dataset are shown in Table 3. This is preliminary work in the direction of using context, as well as feature similarity (saliency), for activity recognition in the proposed tasks.

[A] [Activity Recognition by Integrating the Physics of Motion with a Neuromorphic Model of Perception](http://www.ee.ucr.edu/~amitrc/wmvc09-1.pdf)  
R. Sethi, A. Roy-Chowdhury, S. Ali, IEEE Workshop on Motion and Video Computing, 2009



**Table 3:** KTH Distance Matrix where we highlight the lowest relative values in a row. This shows the matching of similar activities despite view changes with only a few exceptions to correct matching. Please note this is not necessarily symmetric because we do the analysis row-wise using training and classification.

**Related Work:**

**Methods for track based complex activity recognition**

A good amount of research has been done on activity recognition but most of it assumes high resolution data and cannot be extended to lower resolutions. A survey of some of the work on activity modeling and recognition clearly shows that most of the methods are tuned for high resolution data [4, 5, 6, 7, 8, 9, 10]. This is also apparent by analyzing some of the commonly used activity recognition datasets (e.g., Wiezman [4], KTH [10] and IXMAS [15]). The authors in [14] proposed a method for recognizing low-resolution activities by modeling the shape of the tracks of the objects. However, generalization of the approach is difficult since it relies on learning dynamical models to describe the interactions. Learning such models as the number of objects increases is impractical. The research in [2, 16] analyses low-resolution videos but requires periodicity be present in the motion.

Availability of large training sets has been another assumption for developing activity recognition systems. Liu et al [6] propose learning of an optimized codebook for human action analysis. The research work in [3] extracts sparse spatio-temporal features which are used to perform matching across behaviors in video. The authors of [17] learn semantic visual vocabularies of actions by motion feature pruning based on spatial and temporal feature statistics. The use of multiple features for human action recognition is proposed in the work in [7] where seemingly heterogeneous features are embedded in a common graph. All these approaches assume existence of a large number of examples which is impractical for recognizing complex interactions, given the large number of possibilities. We look at how to retrieve activity clips given a single query from the user.

There has been some work on activity recognition based on tracks obtained from the video under consideration. Parameswaran et al [12] extend the cross-ratios for trajectories [13] in two and three dimensional spaces and successfully apply in the domain of human action recognition. This approach does not model the interactive/complex interactions as required in our problem domain. Rao el al [13] learn the spatio-temporal curvature based view-invariant features from the tracks of the hand of actors performing some action out of a predefined list of actions. Their approach also does not model complex/interactive activities between multiple objects. Ali et al [1] learn chaotic invariants from the tracks of body joints which are used to recognize human actions. In addition to the assumption of the tracks of each body joint, it is not easy to generalize their approach to model complex/interactive activities.

**Models for learning dynamical contexts**

The notion of context is not new and has been explored in different areas like linguistics, natural language processing and knowledge representation. But it is still new to the field of activity recognition, and there is not much work on how to learn the contexts in activity recognition. Probably the most related work in this field is the activity pattern learning. However, most of the works focus on learning traffic pattern [18, 19], and not consider the relationship between the targets and contextual objects in the scene. The problem of automatically acquiring context models from data is addressed in [21], by using a situation model to represent context and human behavior. This approach requires the prior knowledge of the situation categories which may not be easy to know given the large number of possibilities. Since the proposed model is applied on short sequences, they don’t look at the statistics of the patterns, which is very important for analyzing the activities during a long time period. Patino et al [20] use two clustering processes to derive the knowledge of people activity and extract the relation between people and contextual objects. Agglomerative hierarchical clustering is used to find the main trajectory patterns of people and relational analysis clustering is employed to extract the relationship between people, contextual objects and events. The relationship between people is modeled in a somewhat simple way and the dynamic characters of people relations are not considered. The study on learning of dynamic social network using statistic tools as importance sampling and EM algorithm [22] is good inspiration. But the methods can not be directly adopted for the dynamical contexts modeling.

**Context aware activity recognition**

Early work on activity recognition was mainly based on the use of data acquired from imaging devices (e.g. cameras) or body-worn sensors. One of the limitation of these work relied on the fact that they did not consider contextual information (such as current location, environmental conditions etc) that could be usefully exploited to derive the activity. The main concern in context aware activity recognition is to find an efficient way to integrate contexts. Researchers have proposed computational frameworks for integration, e.g., [23], but they have also been restricted to the analysis of single images. The use of DDMCMC shown in [24], or its variants, might be an excellent Integration module application. A context-aware activity recognition framework is proposed in [25] with the specification to the application in hospital, which makes its generalization be difficult. In [26], the authors propose the integration of ontological reasoning and statistical inferencing to address the problem of recognizing activities based on contextual conditions. Their method is designed for body-worn sensor data, which is difficult to obtain in most application scenarios.

Paul’s summary:

Toni does not work on feature extraction, focuses on reasoning about co-occurrence of action patterns, e.g. can someone be digging and running at the same time? Uses abstract, stick-like sketches.

This work is strongly related to Damian’s work on schemas. Schemas can be represented in many different ways, and Toni suggests a number of different ones, including HMMs, CFGs, Bayesian networks, etc. I think we can stay away from having to choose one formalism in the proposal, and can open a dialog between Toni and Damian during the project (if we are funded!)

Analytical module.

Toni

The space of possible

REPRESENTATION

the noun (person/object) representation need to have the minimal complexity necessary to model and capture the interactions (verbs) between them.

Action-Primitives, our "symbols" which are simple motion patterns:

In terms of representation of atomic action. Have the model be able to deal with any abstract representation of some single-frame “parameters” changing with time whatever they may be.

Simpler representations are favored.

(extracted "feature" from a few consecutive frames using via available vision algorithm)

These are taken from some other software, NO FOCUS ON THESE

- stick-figure/skeleton of a person

- PCA

- joint angles like motion-capture

- human and object silhouette/outline

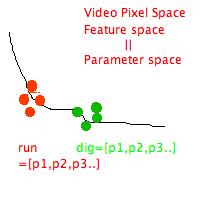
- histogram of the edge orientations

- Fourier shape

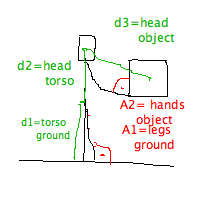
Model Actions: the interactions between Person and Object, Person and Person

A representation should “Map” actions, described by some parameters, to a subspace where all instances of one action map to the same point/cluster and such is far from other

The system maintains symbolic description of the world. Relating these symbols at different time scales creates a system with hierarchical complexity of reasoning. The symbols will represent spatio-temporal patterns or action primitives that are easily discoverable by existing vision algorithms and can be repeatedly extracted without error. Thus, we can assume that our symbols are valid and focus the problem on reasoning about their co-occurrence. Analysis can be performed accurately.



* Store action in a compact representation
  + Break scene into “blob” parts: ground, persons(legs, torso, hands),objects
  + Joint angles, hierarchical from ground, legs up to hands holding object
  + Relative position also hierarchical
  + Change of relative angles and positions (trajectory) over time



* on-line unsupervised learning

USER FUNCTIONALITY

* Monitor for action requested by user
* Retrieve instances of some action from the past
* Animate action for user

INTELLIGENT REASONING

- Learning

A trained soldier will view the same scene very differently than a civilian. They have different idea of the organization of the world. Our system will be modeled and taught to view the world with the eyes of a solder by using a meaningful contextual representation.

Action recognition Methods: learn/recognize/extract

- HMMs

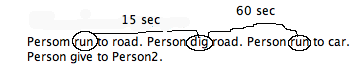
- contex-free grammer

- fuzzy logic

- Bayesian netwrok

- EM

Activity is recognized in state space, where the abstract components, or states, of the activity reside. This enables the recognition of the same activity even when it physical performance is various (there are different ways to "exchange" object). Learn such long-term activities will enable us to predict what will happen by observing part of the activity. Human know from experience that some action often fallows another action. Moreover, we can reason logically about what could have happened when the actions were occluded or unobservable. Such deductive reasoning can be achieved by recognizing components/actions in the beginning and end of a complex activity.

* Reason about composite event, sequence of actions
  + Imagine the “actions” as a sentence, describing what’s happening, that is formed over time. A frequency graph of what actions connect to other actions and how far are actions in time. Also the general subject of the actions (car, person, road) The reasoning will learn the “semantics” of this language of actions. 
* Predict future action, Interpolate/Guess “occluded” actions
  + This fallows from the reasoning
* Detect abnormal activity (not observed before)

Moreover, expansion beyond the initial set of about 50 actions/activities to model larger human behavior renders to this system. One can imagine that the system will be able to describe this behavior in its own language, just like a human observer. Keep in mind, that different observers will give different description of the same scene of events. Thus, by training the system to pay attention to certain terms in the desired context, will produce a system behavior language that is very much in the "style" understood by the trainer (army personnel). For example, the system doesn't need to speak of fashion style when it sees a bulky object under the sweater. Such an expertly trained computer narrator, describe relevant activity in the context.

Atomic action Primitive

Action

Activity

Scenario

Paul’s summary:

Uses process algebras to represent and analyze action. RS schemas have a formal semantics that is an algebra of automata, and process algebras are strongly related to this. Both have been around for decades: process algebras used to analyze concurrent programs, RS in robotics specifically. RS is very similar to CSP. If we are funded, Terry should have a long chat with Damian and Paul.

Does not do extraction directly from video. This fits in the analytical module.

Figure 1 looks good.

Reinforcement learning and decomposed decision trees are the learning approaches proposed.

Terry – actions/activity

Actions/activities are mapped to a Cost-Calculus ($-Calculus) framework [Eberbach, 2005], and observed actions/activities between agents are matched to the existing actions/activities database using well-known bisimulation equivalence relations which are binary relations between state transition systems, associating systems which behave in the same way in the sense that one system simulates the other and vice-versa. The advantages of this approach with regard to the current state of practice:

1. Easy to integrate for testing due to the an existing behavior-based system running on UGVs under CARACaS
2. Reduced computational complexity with respect to existing algorithms (linear vs. polynomial or exponential) leading to efficient onboard use
3. Rigorous mathematical foundation (Process Algebras) that supports analysis
4. First approach to explicitly factor in sensing from a moving platform and analysis of actions by other independent agents in the surrounding environment

This approach has been used successfully in a number of different fields, including motor schemas for robotic control and plan recognition in economic processes.

A Cost Calculus ($-Calculus) [Eberbach, 2005] is a model for resource bounded computation based on process algebras that:

1. Provides a means for generating incremental solutions for computationally hard, real-life problems
2. Provides a uniform representation for the use of uncertain information during the cost-optimization process (kΩ-optimization)
3. Currently is the basis for the CCL (Common Control Language) used for behavior-based control of UUV (Unmanned Undersea Vehicles) [Buzzell, 2004; Duarte, et al., 2005]

Actions/activities are written as $-expressions organized into sets based on context in order to optimize inference operations. $-expressions are built using the algebraic operators of send/receive, cost assignment, defined simple/process call and sequential/parallel composition.

For the inference of sensed actions/activities, the observation equivalence of behaviors on a single autonomous agent and between two or more agents is done through bisimulation relations:

1. Unobservable actions/activities (incomplete knowledge about an agent or the environment) is represented by the silent (invisible) action *ε*
2. Actions/activities are formally expressed as a TLTS (Timed Labeled Transition System)[Milner, 1994; Lee & Zic, 2002], and as such, the bisimulation equivalence can be established in linear time [Paige & Tarjan, 1987]
3. Linear efficiency enables both onboard and/or offboard use of the technique

An example of the inference mechanism at work on an IED placement scenario is shown in Figure 1, where the sequence of actions for placing the IED include *Stepping on Roadway, Walking onto Roadway, Crouching to Place IED, Walking off of Roadway, and Stepping Off RoadWay*. The autonomous vehicle uses its sensory inputs and infers what the actions are being done through an inference and comparison with the alternate sequence of Crossing the Roadway. Sequential composition operators are used for the sequence and actions occurring in the future are represented by the silent (invisible) action *ε* until sensory inputs arrive*.* Sending all of the sensory information back to the operator and relying on her to interpret it rather than just sending a higher level explanation such as “there is evidence that an IED has just been placed at <Position X>” imposes an unnecessary burden on the operator and could lead to misinterpretation.

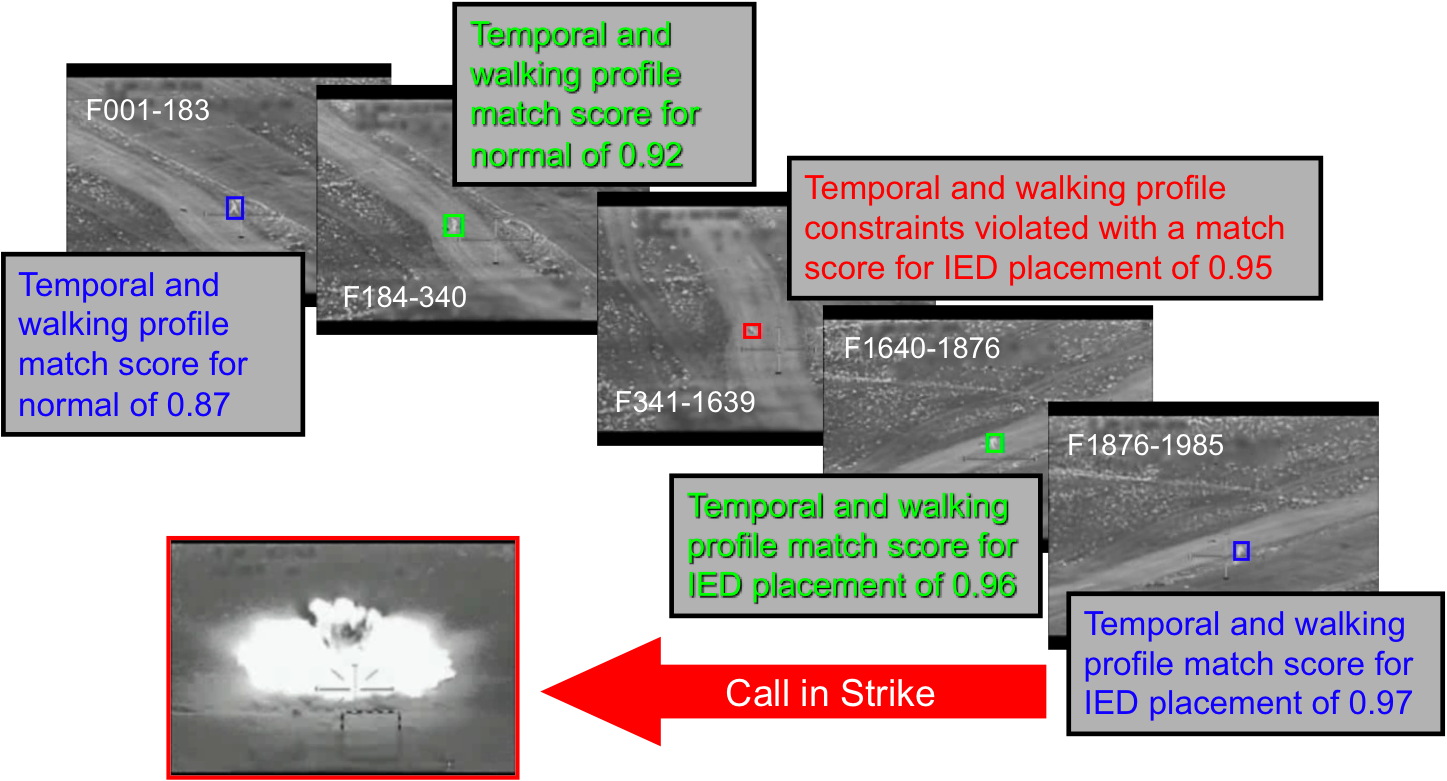


Figure 1. Example of Process Algebra inference progression for IED placement scenario.

For the learning of sensed actions/activities which is necessary for common grounding of activity sequences that were not previously observed or in the command dictionary of the autonomous agent, reinforcement learning of observed action/activity patterns will be used. Reinforcement learning is a mode of the kΩ optimization built into the $-Calculus and:

1. Unobservable actions/activities (incomplete knowledge about an agent or the environment) are represented by the silent (invisible) action *ε*
2. LTS representation used for generation of history of behavior use (similar to networks of Michaud & Mataric, 1999)
3. Q-learning update equation is directly represented in the $-Calculus using cost / general choice and sequential composition operators

Efficient onboard reinforcement learning algorithms have been previously developed at JPL and demonstrated for adaptive behavior in out-door environments on rovers [Huntsberger, Aghazarian, and Tunstel, 2005].

For the development of explanation capabilities, a dynamic decision tree decomposition [Utgoff, Berkman, & Clouse, 1997] of the observed actions/activities is used to generate a set of rules for explanation:

1. Unobservable actions/activities (incomplete knowledge about an agent or the environment) are represented by the silent (invisible) action *ε*
2. Decision tree generation uses information gain and pruning to limit the size of the tree
3. Rules are evaluated based on hit rates, miss rates, pessimistic error rate, and information gain

An adaptive level of detail is automatically built into this process in that all of the sensory information that led to sensed action/activity is available and can be conveyed to the operator if the HMI has a detail level of request capability.

Paul’s summary:

Uses virtual world to model real world. Goal is to use simulated physics to predict actions, then match them to observed actions in real world. Successful match means correct classification.

Uses RS to model actions. RS constructs automata for the dynamics of each component, then connects the automata using ports. The algebra of these automata has been developed and is similar in many ways to process algebra (viewing each automaton as a process.)

Uses Soar cognitive architecture to implement RS, thus adding explanation-based learning.

Some work done on extracting features from video. That part fits in the extraction module. The virtual world is in the analytical module.

A modified version of Figure 1b could be useful for the explanation module.

A few of the figures from the cameras might be useful if detail is needed for the explanation module.

THE FOLLOWING OF THIS SECTION IS UNPROCESSED INPUT FROM PAUL AND DAMIAN – NEEDS TO BE FILTERED AND REFINED/ALIGNED WITH REST OF PROPOSAL

**Some Text on RS and Soar:**

The RS (Robot Schemas) language is a language with a formal model of concurrent robot computation that is based on the semantics of networks of port automata. A port automaton (PA) is a finite-state automaton equipped with a set of synchronous communication ports.

RS builds a network of *sensory-motor schemas* to model the dynamics of both the robot and the environment. A schema is an organized network of actions (abstract and/or concrete), percepts, words, facts and beliefs about some aspect of the world. Schemas also contain explicit qualitative temporal information, e.g. about intervals of time during which an action must take place. The schema relates these components to each other and to other schemas.

For example, the schema for a table would contain various images of tables and words that describe tables, and would connect them to facts about tables, such as their typical size, and to actions that typically are performed with tables, such as sitting and eating at one (which is another schema). The table schema would be connected to the schemas for chairs, dining rooms, conference rooms and many other relevant concepts.

One of the unique advantages of RS is its formal semantics. Each schema has an associated port automaton that defines the semantics of the schema. The PA provides the basic semantics for the use of natural language. States in the PA provide the semantics for relations and predicates (adverbs and adjectives). Transitions between the states provide the semantics for verbs.

RS provides a powerful representational language for the system's dynamics, language and percepts; however, RS does not provide reasoning or learning mechanisms.

We have implemented RS in Soar [10] to take advantage of Soar's cognitively plausible problem-solving and learning mechanisms, as well as the existing work on other aspects of cognition in Soar, including natural language [13,27], concept learning [31], and emotion [22].

Soar uses *universal subgoaling* to organize its problem solving process into a hierarchy of subgoals, and uses *chunking* to speed and generalize that process. We use Soar’s subgoaling ability to create hierarchies of schemas, and uses Soar’s chunking ability to speed the creation of these hierarchies.

Soar manipulates a hierarchy of problem spaces, and the formal semantics of RS consists of a hierarchy of port automata. We have merged these architectures in a straightforward way by implementing each RS schema as a Soar problem space. This is done by specifying the state transitions of each schema's port automaton in Soar.

Merging RS and Soar in this way combines their strengths. The strengths of RS include its formal mechanism for combining sensing and motion, its ability to reason about the temporal behavior of schemas, and its combination of deliberative planning and reactive behavior. Its weaknesses are the lack of a synthesis mechanism for autonomous formation of sensors or actuators, and the lack of a model for implementation of cognitive abilities such as learning and language.

Soar provides an integrated cognitive model with a full range of cognitive abilities, including perception, deliberative planning, reaction, natural language, learning, and emotion. But Soar lacks parallelism and a temporal mechanism, which severely hampers its usefulness in robotics.

Soar RS

Figure 1. Soar problem spaces implement RS port automata.

**Some introductory overview text:**

There is substantial evidence that humans possess innate spatial representation and transformation abilities. Research in cognitive science using subjects who have been blind since birth confirms their ability to reason about 3D representations, and in particular to perform mental rotations and translations [17,18,34,35]. Given the fundamental nature of such a mechanism in humans, we wish to implement it in an artificial vision system.

The goal of our project is to build a robot vision system that comprehends visual data by creating a working virtual model of itself and its environment. This virtual world can be used to visualize scenes from multiple perspectives, and the simulated physics of the virtual world can be used to predict possible future behaviors.

Our system models objects as hierarchical automata, and models their dynamics as concurrent state transitions within this hierarchy. Dynamics are recognized by matching observed behaviors with the predicted behaviors generated in the virtual world. The virtual world acts as a visual imagination, literally a minds eye, for the robot system. The spatiotemporal events of known actions, represented as schema networks, influence both visual attention and also virtual simulation, giving rise to visual expectations that can be compared directly with visual experience. Dynamics of actions and events are learned from experience and expectations using known techniques for inductive inference of automata.

We are using the RS language (Lyons, 1987) to represent spatiotemporal patterns of behavior. Each local pattern is called a schema , and RS composes schemas using a full set of concurrent combinators. Although concurrency is not explicitly included in the program call, we believe concurrency is necessary to model even simple behaviors. Two or more actions often occur simultaneously, e.g. turn and lift, or follow and carry. Also, a single action is often implemented in terms of others, e.g. opening by kicking, or replacing by lifting. As a result, a visual recognition system will often recognize multiple actions simultaneously, and must be able to represent and reason about concurrent behaviors. Failure to include concurrent representations would limit a recognition system.

Schema representations are task-oriented in the sense that the objects that play a role in the task, the 'nouns' are defined not just in terms of appearance but in terms of thir role within the task - the perceptual

schema. This is an essential part of interpreting actions in their broadest sense and it sees objects as simply aspects of the environment that yield affordances for action.

A major goal of our project is to recognize and reason about human behaviors. This requires the system to employ a model of human cognition, so that it can reproduce human behaviors. We achieve this by implementing RS within the Soar cognitive architecture (Laird, 1987). RS provides the representation of concurrent behaviors, but lacks reasoning and learning capabilities, which are provided by Soar. Soar is a mature computational model of human behavior, encompassing symbolic reasoning and learning abilities. The implementation is straightforward: Soar creates and modifies the automata that correspond to RS schemas. Soar's built-in learning method permits it to learn new automata from examples.

Envisionment is a built-in aspect of our approach, since schema network representing spatiotemporal patterns direvctly produce visual simulations. While at this point our current visual output is low-level and explicit, in the proposed work we will produce envisonments that are higher level and more abstract.

**Some text on our virtual world:**

The growth of the video game industry has contributed to the ready availability of good 3D simulation environments that can create a wide variety of virtual worlds, and possess sophisticated graphics and realistic physics that run in real time. These simulators can model multiple independent agents acting concurrently, and also provide the ability to view the virtual world from any point in any direction.

These recent advances in video game software have led a number of projects to integrate robot simulation and control environments, e.g., [1,19]. Such approaches are focused on providing ready interleaved access to simulation or robot control to a programmer. In contrast, our use of simulation is as an intrinsic part of the robot control itself, not as an aid to the programmer. In the assembly and task planning field, the integration of simulation into the reasoning process has been investigated [39]; however, this was achieved by integrating the planning and simulation software modules, sharing a common data structure. Our objective is to integrate 3D simulation in such a way that the robot sensing, control, planning and learning modules don’t share data structures directly with the simulation. The choice and selection of data structures for the simulation – how objects are represented, how physical effects are calculated, etc. – will impose constraints on the design of robot sensing, control, planning and learning architectures. Our objective is for the interface between the robot architecture and the simulation to be as close as possible to the interface with the physical environment, creating a situation where the simulated and real worlds can be interchangeable and intermixed.

In the current implementation, our world model is the Ogre3D open source gaming platform (http://www.ogre3d.org). OGRE (Object-oriented Graphics Rendering Engine) provides the ability to create a detailed and dynamic virtual model by providing sophisticated graphics and rendering capabilities together with a physics engine based on the PhysX physics engine. The PhysX engine is very fast and accurate, and if more speed is necessary hardware acceleration can be added.

Ogre is capable of modeling a wide variety of dynamic environments including other agents moving and acting in those environments. Our virtual world combines the various forms of map information found in most robots: topological, metric and conceptual information. One of the most useful abilities of Ogre is that it can create and move virtual objects very quickly. This is very handy when modeling newly perceived objects in the virtual world or correcting the position of virtual objects.

The Ogre community provides a number of example worlds that include models of many objects and of people. We have constructed worlds that include models of our Pioneer robots, and have written code that renders simple objects into these virtual worlds. Object recognition code from the Intel OpenCV Library is used to identify objects. Virtual copies are then retrieved from a library and positioned in Ogre. RS structures point to their Ogre counterparts and are continually updated by Ogre as they move. Some videos of rendering visual data into a virtual world are available at <http://csis.pace.edu/robotlab/movies.html>

**Reasoning/planning**

(object ^type chair ^id 143 …)

(object ^type wall ^color black ^id 122)

Object Recognition

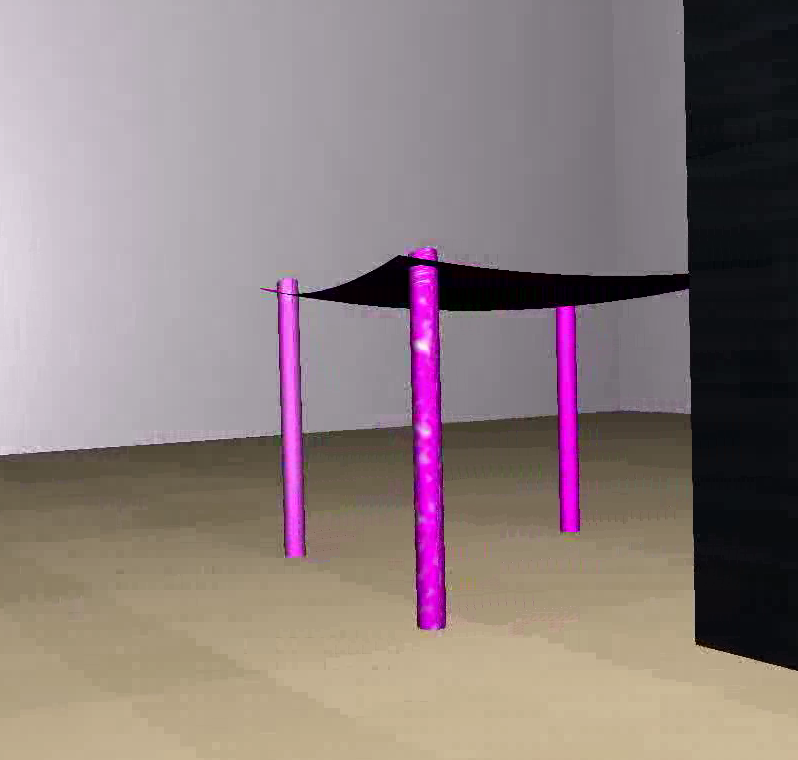
 

Figure 2. Objects, relationships and dynamics are modeled and annotated in the virtual world.

**Some text on our predictive approach to vision:**

As an illustration of the use of the virtual world, let us consider *predictive vision*, a novel method for quickly locating and tracking objects with complex behaviors. Predictive vision leverages our 3D simulation engine to generate predicted imagery and compare that against real imagery from the cameras. Predictive vision is based on a novel technique, the match-mediated motion difference, for comparing real and synthetic images that takes into account that the two images may be taken from different camera viewpoints, may contain some differences in color and texture, and may contain different objects. In this approach, the salient points in real and predicted images are identified and an affine image transformation that maps the real scene to the synthetic scene is generated. An image difference operation is developed that ensures that the matched points in both images produce a zero difference. In this way, synchronization differences are reduced and content differences enhanced.

Significant content differences are used by the application to select actions. For example, a content difference in an autonomous car’s vision system may be a new car in its environment. The car must determine whether the new vehicle will intersect its path, forcing it to re-plan.

Predictive vision greatly reduces the cost of vision. As long as the behavior of the environment is sufficiently similar to the modeled behavior of the virtual world, then content differences will remain small and are ignored. In this case, no expensive visual processing is required. Significant discrepancies mean that the virtual model needs to be updated, either because an object has appeared or disappeared, or because an object’s behavior has changed. In this case, visual processing can be restricted to the part of the visual data containing the discrepancy.

Figure 1b: Block diagram of the loop integrating simulation and observation

Predictive vision requires comparing the synthetic and real imagery to look for differences between actual and predicted object behavior. Comparing the synthetic imagery from the world model with imagery from the vision system poses a number of problems [16], including synchronization differences: differences in the camera poses, in the scene lighting, and in the colors and textures; as well as content differences: differences in the number and type of object shown and differences in the predicted object behavior. All these issues mean that a simple difference operation between real and synthetic images is not very effective.

Figure 1b shows the block diagram of our overall system [6]: The real and synthetic images of the scene as viewed by the robot are compared. If the scenes are considered the same but from different viewpoints, then the viewpoint of the camera in the simulation is changed, and the simulation generates an image taken by the camera at the new location. If an unexpected object is seen in the real image, an object is introduced at the corresponding position in the simulated scene. The region of the real image responsible for the difference is used as video texture on the object and a new synthetic image generated. The information on whether there is no difference, an unexpected object, or an object missing between the image pairs is made available to action planning [5]. This loop of difference detection and simulation modification is used to keep the simulation synchronized to the observed environment. For prediction purposes, the simulation can be allowed to ‘fast forward’ in time, so that the expected position, for example, of a target can be calculated and then compared to observations.

Fig. 2 shows a real (2(A)) and synthetic (2(B)) view of the same scene taken with the artificial camera at approximately the same location and orientation as the camera in the real scene.

(A) (B)

Figure 2: Real (A) and synthetic (B) views of the same scene from approximately the same position and orientation

The view presented here is a close up of one wall of a room. The scene is also modeled graphically using OGRE. Sections of the graphical scene have been tiled with video texture manually extracted from Fig. 2(A). The use of video texture should make it easier to directly compare the real image and synthetic image to answer the following questions:

1. Do they represent the same scene from the same viewpoint?
2. Do they represent the same scene from slightly different viewpoints?
3. Do they represent the same scene but with some number of different objects?
4. Do they represent different scenes?

Let *Is* be the simulated image and *Ir*be the real image. Fig. 3(A) shows the absolute image difference | *Is- Ir|* and Fig. 3(B) is a thresholded absolute difference. The images show substantial differences because the real and simulated camera positions are not identical and errors are introduced by the texture extraction and tiling. However, we would like to be able to determine that these are views of the same scene, albeit from slightly different camera positions.

(A) (B)

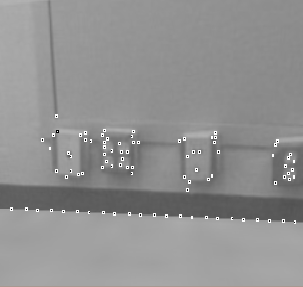
Figure 3: Absolute difference (A) and thresholded difference between real and synthetic images from Fig. 2.

**Alignment of Synthetic and Real Images**

If we just consider the camera misalignment issue, we can approximate the registration between the real and synthetic images by an affine transformation. If *ps=(xx , ys)*  is a point on *Is* and *pr=(xr , yr)* is a point on *Ir* then we can say:

*pr = A ps + b* (1)

where *A* is a *2 × 2* rotation matrix and *b* a translation. If points of correspondence can be established between the real and synthetic images, then the affine parameters *A* and *b* can be estimated. Using the efficient corner detection library of Trajkovic & Hedley [38], corners were labeled in the real and synthetic images (e.g., Fig. 4(A, B) for the scene of Fig. 2). The RANSAC algorithm [19] was used to estimate the affine parameters *A* and *b.* Fig 4(C) shows the real image transformed for registration with the synthetic image, *I’r* and Fig. 4(D) shows the points used to estimate the transform.

(A) (B) (C)

(D) (E) (F)

Figure 4: Corners detected in real (A) and synthetic (B) scenes, real scene affine warped to synthetic scene (C) using matched points (D), absolute difference (E) and thresholded difference (F) with warped real and synthetic images.

The difference operation | *Is- I’r|* is shown in Fig. 4(E) and the thresholded result in Fig. 4(F). Comparing Fig. 3(B) and Fig. 4(E), some of the sources of the difference error have been resolved but not eliminated. There is less difference error on the lines and edges on the wall and floor, but the affine registration on its own is not sufficient.

**Calculating the Match-Mediated Difference Mask**

The affine transformation brings corresponding objects in the real and synthetic images approximately into registration. However, there are still differences caused by the quality of the texture mapping or simulated surface color. To address this, we will make the assumption that the image area around a point used to estimate the affine registration should be similar in both images. The better two matched points correspond between real and synthetic images, the more we will assume the two images should be similar.

Let *pr* and *ps* be two matched points in the real and synthetic images. We will consider the synthetic image to be the primary image and construct the difference image in those image coordinates. Each match point *p’* in the set of match points *P* will be in the image coordinates of the synthetic image. Its corresponding point in the real image, *m(p’)*, will be given by the affine transform in eq. (1):

*m( p’ ) = A p’ + b* (1)

We will place a normalized Gaussian at each point *p’* in the set of match points *P* and sum these over the image to create an image mask whose values correspond to the proximity of the image pixel to adjacent match points:

** (2)

where is the sum of the Gaussian for *p’* over the entire image *I* and *v* a small fixed variance parameter:

 (3)

However, this doesn’t account for the fact that some matches are of better quality than others. If *p* is a pointin the set of match points *P*, we define the match error *e( p )* to be the distance between the two matched points *p* and *m(p)* in the primary image coordinates:

*e(p) = | p– m( p ) |*

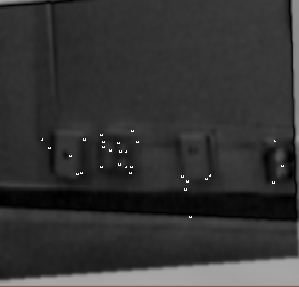
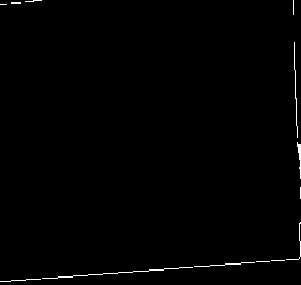
We define the normalized match quality *q(p)*  to be the inverse of the match error normalized by the sum of all match errors:

 (4)

Eq. (4) is a measure of the quality of match *p* with respect to all the other matches, and we use this as a coefficient for the Gaussians in Eq. (2) to generate the *match-mediated difference mask Im :*

** (5)

Figure 5(A) shows the match points for the running example superimposed on the absolute difference image| *Is- I’r|*. Fig. 5(B) shows the resultant gray level match-mediated difference mask.

(A) (B) (C) (D)

Figure 5: Absolute difference of warped real and synthetic images with overlaid match points (A) and match mediated difference mask (B) calculated from (A). Match mediated absolute difference (C) and thresholded difference (D).

To calculate the match-mediated difference image using the match-mediated difference mask eq. (5) we divide each point in the difference image by the corresponding point in the mask:

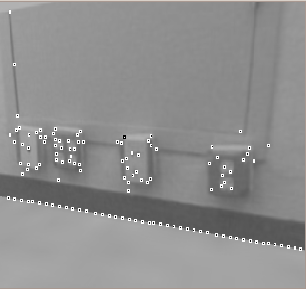
 (6)

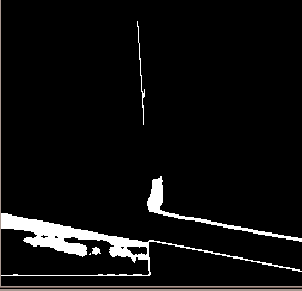
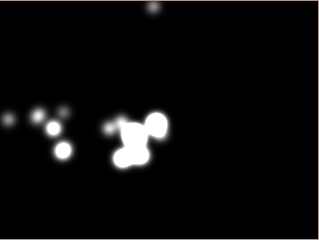
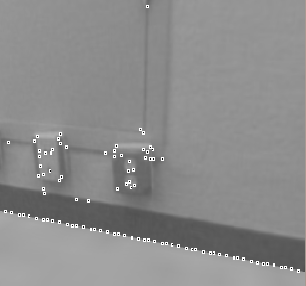
The result for the running example is shown in Fig. 5(C) and thresholded in Fig. 5(D). (Thresholds used throughout are the same for all images). The resulting difference image shows only the edge of the common region in the primary (synthetic) and affine transformed secondary (real) image, so we conclude that both images are of the same scene from different viewpoints as given by *A* and *b* from eq. (1).

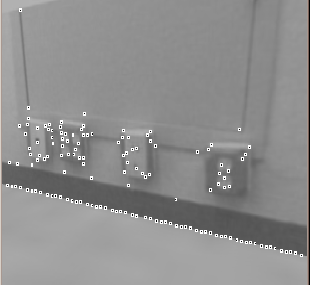
**Experiments**

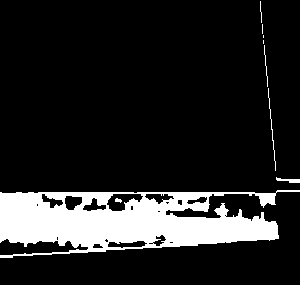
In this section we show the results of experiments using the match-mediated difference approach to detect whether real and synthetic images show the same scene, a scene with a new object, or a scene with a missing object. Fig. 6 shows four pairs of real and synthetic images. The first three pairs attempt to match the same real scene with slightly different views of the synthetic scene. The first and fourth pair attempt to match the same synthetic scene with slightly different views of the real scene. In each case, the image pair is shown in columns (A) and (B) overlaid with the corner point results; column (C) shows the affine transformed real image; column (D) shows the gray-level match-mediated mask; and, column (D) shows the thresholded match-mediated absolute difference image. In the final column, the only part of the difference image that is valid is the overlap between the synthetic and affine transformed real image and the boundary is typically visible.

The first image pair shows the ideal result, the difference image is empty in the overlap region. However, in the remaining three rows, the image is blank for most of the overlap region, except for the floor. The region of the image with the most complicated geometric features remains blank because of the math-mediated difference mask. The floor in the synthetic scene is visually quite different from the floor in the real scene, the result of imperfect manual texture collection from the real scene and mapping in the simulated scene. However, the image scope of the difference is sufficiently small (unlike the large areas of difference in Fig 3(B)) that we expect that we will be able to use this to iteratively refine the texture in the simulated scene and reduce the observed difference via the loop shown in Fig. 1b.





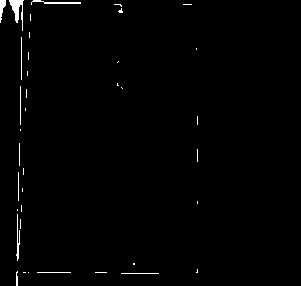




(A) (B) (C) (D) (E)

Figure 6: Examples of the scene in Fig. 2 but with the simulated camera moved: Cols. A and B are the real and synthetic images with corner points; C is the affine transformation image; D is the match-mediated mask; and, E the match-mediated difference.

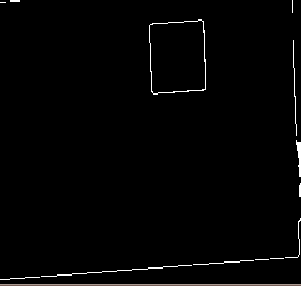
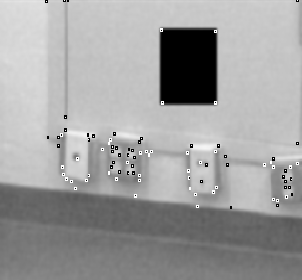
All the examples so far are of image pairs that should produce no difference. To be useful, this approach should preserve differences that are due to new objects in either real or synthetic image. Our convention is to consider an object in the real image but not in synthetic image as an *unexpected object*, and an object in the synthetic image but not the real image as a *missing, expected object*. In Fig. 8, the top line shows the same experiment presented as the running example in Section 3, except a black square has been artificially drawn on the back wall of the real image. The process of

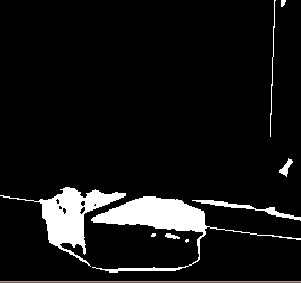
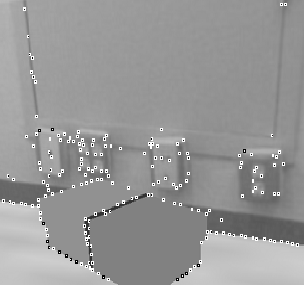


(A) (B) (C) (D) (E)

Figure 7: Book example images: Cols. A and B are the real and synthetic images with corner points; col. C is the affine transformation image; col. D is the match-mediated mask; and, E the match-mediated difference.

estimating the affine transform is the same as for the original experiment. However, since there are no matches on the black square – as it appears in only one image of the image pair – the match-mediated difference mask contains zero or small values in the vicinity of this feature (Fig. 8(D)). Hence the feature is preserved when the difference operation (eq. (6)) is evaluated, showing up clearly in Fig. 8(E).

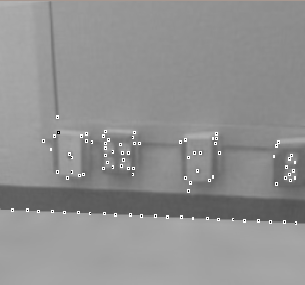
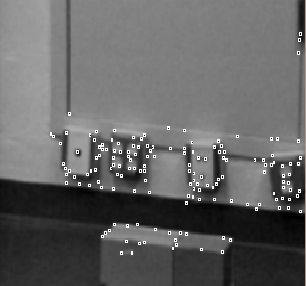




(A) (B) (C) (D) (E)

Figure 8: Example of a scene with an expected object missing: object is in synthetic image and not in real image: Cols. A and B are the real and synthetic images with corner points; col. C is the affine transformation image; col. D is the match-mediated mask; and, E the match-mediated difference.

The second row of Fig. 8 shows a box introduced into the synthetic scene. The scene was generated by making a graphical model of a box roughly similar in appearance to the box in Fig. 9(A) and placing on the floor close to the wall in the 3D Ogre scene model. Because of the proximity of the box to corner features used as match points, the match-mediated difference mask does partially overlap the region of the image where the box is. Nonetheless, the thresholded result extracts the majority of the box as a valid difference region. In this case, we would consider this a missing expected object.



(A) (B) (C) (D) (E)

Figure 9: Example of a scene with an unexpected object - object is not in synthetic image but is in real image: Cols. A and B are the real and synthetic images with corner points; col. C is the affine transformation image; col. D is the match-mediated mask; and, E the match-mediated difference.

Figure 9 shows an example of an unexpected object. The corner points on the box contribute minimally to the affine transform and to the match-mediated difference mask. The thresholded result does indeed show the box against the floor; however, so much of the floor also shows up that it is difficult to identify the box as an expected but missing object. Our approach here, as in the last few examples in Fig. 6, is to use the difference region extracted as a mask to extract floor texture from the video via the loop shown in Fig. 1b. With better floor texture, we expect the box to be separable.

This approach to predictive vision works if a sufficient number of corner points can be extracted from each image and an affine transform can be found to match the images. In the case that an affine transformation cannot be found, the images are considered too different to compare. Another constraint is that real regions of difference are sufficiently distinct from the points used to make the affine transform. This constraint may result in the edges of objects being clipped, as for example in Fig. 8(E) second row.

All the examples here started with a manual extraction of texture for the simulation. A major avenue of future work in this project is automating the loop in Fig. 1b for updating the simulation by extracting texture from regions identified as difference regions. For example, the floor in Fig. 9(E) would be identified as a difference, the difference region used as a mask to extract texture from the real video, and the texture added in to the simulation. This loop should converge by incremental identification of differences, extraction of texture, and updating the simulation model to a zero difference image.

The coupling of the physical world and the virtual world in our system is based on the predictive vision mechanism described in the previous section. Differences between perceived and expected visual input cause the system to update the behaviors or positions of known objects or to create new objects. Behaviors are rendered in the virtual world, i.e. if a ball is rolling in the real world, then the virtual ball rolls in the virtual world. The virtual ball’s position is not simply updated based on visual input; it has a simulated velocity.

This permits the virtual world’s physics plug-in to simulate events in the real world, e.g. collisions. The virtual world software we are using is capable of running much faster than real time. This is helped by temporarily turning off the graphics rendering, which speeds up the software considerably. The system can then predict behaviors for a brief interval into the future, e.g. whether a ball will bounce off a wall or a car will hit a person, and use that prediction before the event occurs in the real world. This should lead to significant improvements in application performance.

In related work with Deryle Lonsdale of Brigham Young University, we have also used this virtual world to provide semantics for natural language [5]. Thus, this approach has a great deal of promise for a range of applications. However, this is a complicated problem, and we face a number of technical hurdles.

Currently, our system uses a fixed library of known objects, and is not autonomously able to handle new objects or textures. In addition, our system cannot handle modified versions of known objects. Nor can it recognize unknown behaviors. Although it seems feasible to engineer useful application libraries by hand, it is clear that a robust system needs to be able to handle real variation in object shape and texture and in behavior parameters.

Towards this end, we are beginning a research effort to render new objects in the virtual world by stretching mesh over keypoints. We are also beginning to look at how to copy textures from the real cameras to the virtual world.

There are also clear limitations to the ability of virtual worlds to model the real world, e.g. simulated lighting and shadows are very different from real ones. We are always wrestling with the issue of how to ignore such differences while extracting significant content differences. This is the central goal of our work. A main reason that we use an open source virtual world rather than the Microsoft Robotics Studio is that we can modify code as needed to help us achieve this goal.

**Some text on RS:**

Our objective in using a virtual world is to model complex behaviors in the environment in such a way that many of the advantages of a behavior based control approach can be maintained, thereby simplifying the control. Modeling static environments or slowly changing environments is not useful for realistic applications, which must be able to model the dynamics of rapidly changing environments, including behaviors of multiple agents acting concurrently and the interactions between these behaviors.

Representing behaviors in complex, dynamic environments requires a powerful and general language for concurrent real-time activities. We are using the RS language [21-25], developed by Damian Lyons. The RS language has been used in industrial applications, including kitting robots [22]. It is a well tested, mature language capable of specifying the full range of concurrent, distributed, real-time programs, and has the power required to represent complex, dynamic environments.

We use RS as the high-level process description language. We have implemented RS in Soar [20], a cognitive architecture. This gives us the ability to use Soar’s RETE matching algorithm to use pattern-matching to invoke relevant RS schemas. In addition, Soar possesses a natural hierarchical structure (called universal subgoaling) that provides a straightforward implementation of RS hierarchies. This implementation permits us to identify the behavior of an object by generating its possible behaviors in the virtual world and comparing their predicted position and orientation with the observed position and orientation of the object over a short interval of time. This is matched using the RETE algorithm against a library of known behaviors to find the correct match. Currently this library of behaviors is hand-built.

**Fodder Material From Demetri Terzopoulos**

Paul’s summary:

Lots of work on modeling humans in a virtual environment

Focuses on modeling individual and collective behaviors using probability theory and a cognitive model

Uses one fixed environment: train station

Not using video, but using a hierarchical world model that encodes perceptual info

An important piece of the overall project: provides a virtual world and well-modeled virtual people

Part of the analytical module

No learning

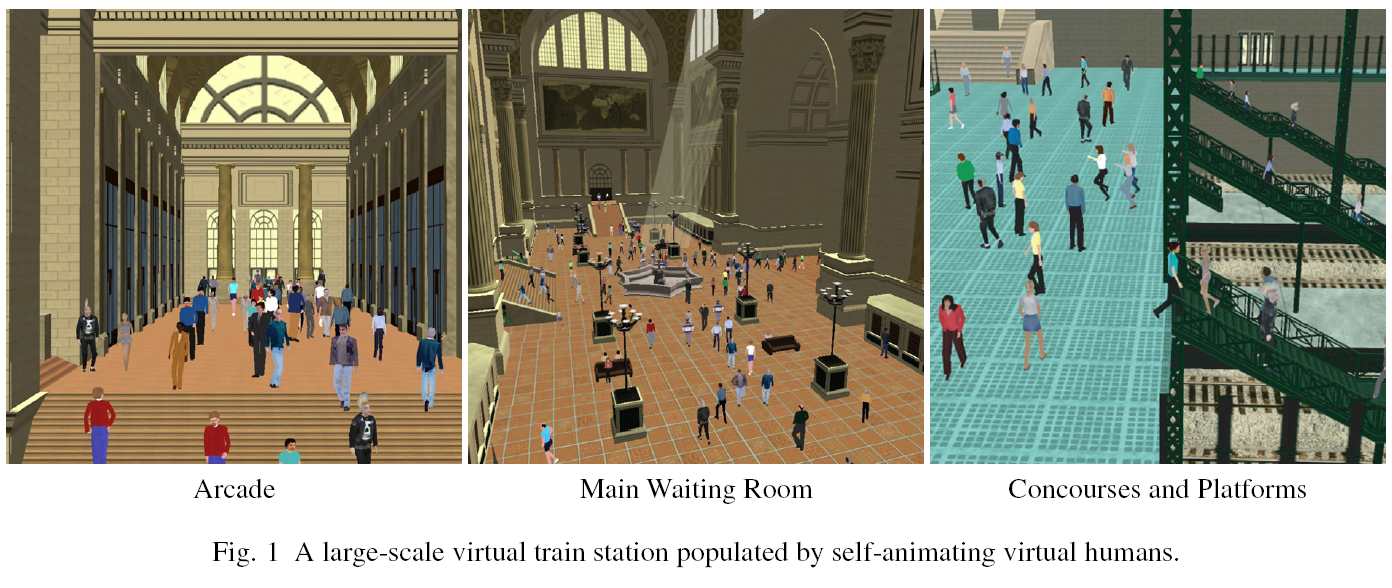
The UCLA team is a world leader in human modeling and simulation. We have done research spanning from the detailed, physically-accurate, biomechanical modeling of the human body to advanced multi-human simulation within an artificial life framework [Terzopoulos, 2009].

In particular, we have developed a comprehensive biomechanical model of the human body that confronts the combined challenge of modeling and controlling more or less all of the relevant articular bones and skeletal muscles, as well as simulating the physics-based deformations of the soft tissues [Lee, Sifakis, and Terzopoulos, 2009]. More relevant within the scope of the proposed project, however, is our work on multi-human simulation (initiated through a 2003 DARPA seedling grant: NBCH1030005) in which we have tackled the problem of emulating the rich complexity of human appearance and activity in urban environments, resulting in *autonomous pedestrian* models that are based on the integration of behavioral and cognitive components [Shao and Terzopoulos, 2007].

**Autonomous Virtual Pedestrians:**

In a departure from the literature on so-called “crowd simulation”, we have developed a decentralized, comprehensive model of pedestrians as autonomous *individuals* capable of a broad variety of activities. Our artificial life approach integrates motor, perceptual, behavioral, and, importantly, cognitive components, yielding results of unprecedented fidelity and complexity for fully autonomous individual and multi-human simulation in large-scale synthetic urban spaces.

Analogous to real humans, our autonomous pedestrians perceive the virtual environment around them, analyze environmental situations, exhibit natural reactive behaviors, and proactively plan their activities. We represent the environment using hierarchical data structures, which efficiently support the perceptual queries that influence the behavioral responses of the autonomous pedestrians and sustain their ability to plan their actions over local and global spatiotemporal scales in an extensive urban environment, currently a well-populated virtual train station [Shao and Terzopoulos, 2005].



As an initial implementation of the appearance and motor levels of the character, we employed a human animation software package called "DI-Guy", which is commercially available from Boston Dynamics Inc. DI-Guy provides a variety of textured geometric human models together with a set of basic motor skills, such as strolling, walking, jogging, sitting, etc. DI-Guy is intended for manually scripted animation. Emulating the natural appearance and movement of humans is a highly challenging problem and, not surprisingly, DI-Guy suffers from several limitations, mostly in the kinematic control of human motions. To ameliorate the visual defects, we have customized the motions of DI-Guy characters and have implemented a motor control interface to hide the details of the underlying DI-Guy kinematic layer from our higher-level behavior modules, enabling the latter to be developed more or less independently.

In a highly dynamic virtual world, an autonomous intelligent character must have a keenly perceptive regard for the external world in order to interact with it effectively. The hierarchical world model is used extensively by each autonomous pedestrian to perceive its environment, providing not only the raw sensed data, but also higher-level interpretations of perceived situations that are important to a pedestrian.

The purpose of realistic behavioral modeling is to link perception to appropriate actions. We adopt a bottom-up strategy [Terzopoulos, 1999], which uses primitive reactive behaviors as building blocks that in turn support more complex motivational behaviors, all controlled by an action selection mechanism.

At the lowest level, we developed six key reactive behavior routines that cover almost all of the obstacle avoidance situations that a pedestrian can encounter. The first two are for static obstacle avoidance, the next three are for avoiding mobile objects (mostly other pedestrians), and the last one is for avoiding both. Given that a pedestrian possesses a set of motor skills, such as standing still, moving forward, turning in several directions, speeding up and slowing down, etc., these routines are responsible for initiating, terminating, and sequencing the motor skills on a short time scale guided by sensory stimuli and internal percepts. The routines are activated in an optimized sequential manner, giving each the opportunity to alter the currently active motor control command (speed, turning angle, etc.).

While the reactive behaviors enable pedestrians to move around freely, almost always avoiding collisions, navigational behaviors enable them to go where they desire. We developed several such routines—*passageway selection*, *passageway navigation*, *perception guided navigation*, *arrival-at-a-target navigation*, etc.—to address issues, such as the speed and scale of online path planning, the realism of actual paths taken, and pedestrian flow control through and around bottlenecks. Furthermore, to make our pedestrians more capable, we have augmented their behavior repertoires with a set of non-navigational, motivational routines, such as *select an unoccupied seat and sit*, *approach a performance and watch*, *queue at ticketing areas and purchase a ticket*, etc.

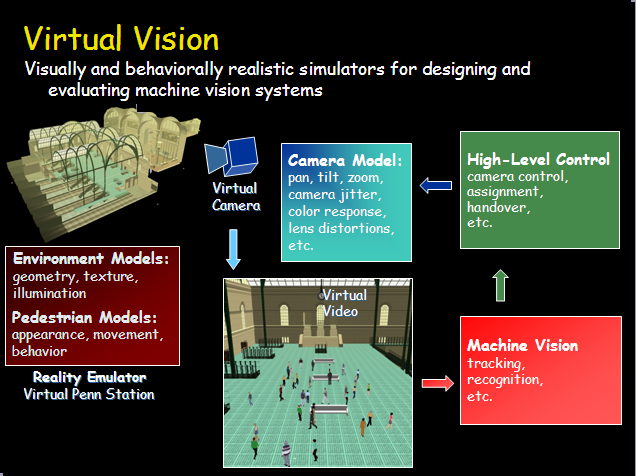
An action selection mechanism triggers appropriate behaviors in response to perceived combinations of external situations and internal affective needs represented by the mental state. For example, in a pedestrian whose thirst exceeds a predetermined threshold, behaviors will be triggered, usually through online planning, to locate a vending machine, approach it, queue if necessary, and finally purchase a drink. In case more than one need awaits fulfillment, the most important need ranked by the action selection mechanism receives the highest priority. Once a need is fulfilled, the value of the associated mental state variable decreases asymptotically back to its nominal value. We instantiate different classes of pedestrians suitable for the train station environment, with each class having a specialized action selection mechanism, including commuters, tourists, law enforcement officers, and buskers.

We have recently developed a decision network framework for behavioral human animation [Yu and Terzopoulos, 2007], leading to advanced strategies for action selection. Combining probability, decision, and graph theories, our probabilistic framework addresses complex social interactions between autonomous pedestrians in the presence of uncertainty. It enables behavior modeling and intelligent action selection subject to a multitude of internal and external factors in the presence of uncertain knowledge, yielding autonomous characters that can make nontrivial interpretations of social situations and arrive at rational decisions dependent upon multiple considerations. We have demonstrated our decision network framework in behavioral animation scenarios involving interacting autonomous pedestrians, including an elaborate emergency response simulation.

At the highest level of autonomous control, a cognitive model yields a deliberative autonomous human agent capable of applying knowledge to conceive and execute intermediate and long-term plans. A stack memory model enables a pedestrian to “memorize”, “update”, and “forget” chains of goals. The stack couples the deliberative intelligence with the reactive behaviors, enabling a pedestrian to achieve its goals. For example, a commuter can enter the station, with the long-term goal of catching a particular train at a specific time. The cognitive model divides this complex goal into simpler intermediate goals, which may involve navigating to the ticket purchase areas to buy a ticket (which may involve waiting in line), navigating to the concourse area, possibly purchasing a drink if thirsty, sitting down to take a rest if tired, watching a performance if interested, meeting a friend, and/or navigating to the correct stairs and descending to the proper train platform when the time comes to board a train.

**Virtual Vision:**

The UCLA team also has extensive experience in combining computer vision and computer graphics in analysis-by-synthesis and synthesis-by-analysis approaches. Our most recent example is a paradigm that we call “Virtual Vision”, which exploits the simulation of the virtual train station populated by autonomous, lifelike virtual pedestrians [Qureshi and Terzopoulos, 2008, 2006]. Active (PTZ) and passive (FOV) virtual cameras were deployed in the virtual train station to capture synthetic video feeds of the virtual human activity, emulating video acquired by real surveillance cameras monitoring public spaces. We can thus design large-scale visual sensor networks in virtual reality, develop multi-camera coordination and scheduling algorithms to control these networks, and experiment with these systems on commodity personal computers. Our approach offers wonderful rapid prototyping opportunities with significantly greater flexibility during the design and evaluation cycle, thus expediting the scientific and engineering process.



In particular, despite its sophistication, our simulator runs on high-end commodity PCs, thereby obviating the need to grapple with special-purpose hardware and software. Unlike the real world, 1) the multiple virtual cameras are very easily reconfigurable in the virtual space, 2) we can readily determine the effect of algorithm and parameter modifications because experiments are perfectly repeatable in the virtual world, and 3) the virtual world provides readily accessible ground-truth data for the purposes of camera network algorithm validation. It is important to realize that our simulated camera networks always run online in real time within the virtual world, with the virtual cameras actively controlled by the computer vision algorithms. By suitably prolonging virtual-world time relative to real-world time, we can evaluate the competence of computationally expensive algorithms, thereby gauging the potential payoff of efforts to accelerate them through efficient software and/or dedicated hardware implementations.

APPENDIX

I move material here in order to make the proposal pages less crowded and try to get into page count. Feel free to use and put back wherever is relevant.

|  |  |  |
| --- | --- | --- |
| Macintosh HD:Users:toni:MindsEye:New:live-bg2:Picture 1.png | Macintosh HD:Users:toni:MindsEye:New:live-bg2:Picture 2.png | Macintosh HD:Users:toni:MindsEye:New:live-bg2:Picture 3.png |
| Macintosh HD:Users:toni:MindsEye:New:live-bg2:Picture 4.png | Macintosh HD:Users:toni:MindsEye:New:live-bg2:Picture 5.png | Macintosh HD:Users:toni:MindsEye:New:live-bg2:Picture 7.png |
| Macintosh HD:Users:toni:MindsEye:New:live-bg2:Picture 8.png | Macintosh HD:Users:toni:MindsEye:New:live-bg2:Picture 9.png | Macintosh HD:Users:toni:MindsEye:New:live-bg2:Picture 10.png |

|  |  |  |
| --- | --- | --- |
|  | Picture 1 | Picture 2 |
| Picture 3 | Picture 4 | Picture 5 |
| Picture 7 | Picture 8 | Picture 9 |

|  |  |  |
| --- | --- | --- |
|  |  | Picture 1 |
| Picture 2 | Picture 3 | Picture 5 |
| Picture 6 | Picture 7 | Picture 9 |

==========POC

JPL

Adrian Stoica: 818 354 2190. Cell: 818 642 6923

Terry Huntsberger: 818 354 5794 Cell: 818 648 0645

POC physically present at JPL for next week (Adrian and Terry on travel)

Toni Ivanov: Office 818 354 5017, Cell: 917 783 8072

UC Riverside

POC: Amit Roy Chowdhury: 951 827 7886 Cell: 626 862 3318

UCLA

POC: Alex Vasilescu, cell: 646-246-5773

Demetri Terzopoulos: ph: 310-206-6946

Pace/Fordham

POC: Paul Benjamin (Pace) 908-625-5782.

Damian Lyons (Fordham): 718 817 4485, cell: 845 214 8797