

# Tracking Objects using Recognition

Randal C. Nelson

Isaac A. Green

Computer Science Department

University of Rochester

Rochester, NY 14627

E-mail: {nelson, igrreen}@cs.rochester.edu

## Abstract

*Tracking is frequently considered a frame-to-frame operation. As such, object recognition techniques are generally too slow to be used for tracking. There are domains, however, where the objects of interest do not move most of the time. In these domains, it is possible to watch for activity in the scene and then apply object recognition techniques to find the object's new location. This makes tracking a discrete process of watching for object disappearances and reappearances. We have developed a memory assistance tool that uses this approach to help people with slight to moderate memory loss keep track of important objects around the house. The system is currently deployed in a prototype smart home.*

Support for this work provided by NSF Grant IIS-9977206, NSF Grant EIA-0080124, and the W.M. Keck Foundation

## 1 Introduction

Tracking can be considered to be the problem of answering the question where is a particular object at a given point in time. The answer is generally expected to be low-level information, e.g. pixels in the image belonging to the object, possibly connected into a compact blob, coordinates of the centroid, possibly with pose information, or location in world coordinates.

Traditionally, tracking has been considered as a frame-to-frame operation[2]. Recognition techniques are often too slow to be directly useful in this domain. However, there are domains in which the objects of interest remain stationary most of the time, for example, keeping track of the location of useful objects around the house, such as glasses, car keys, and the TV remote control. In this domain, using a traditional tracker would be both a waste of resources while the objects were stationary, and unreliable when they were moving, as such objects are typically moved by peo-

ple, and are thus partially or totally occluded during most of the move.

An alternative approach in such a domain is to watch for activity in the environment, detect regions of change that might represent objects added to or removed from the scene after activity has ceased locally, and send images containing changed regions to an object recognizer for evaluation. Tracking an object then becomes a discrete process, with disappearances and reappearances reported by cameras in various locations being integrated into an object location/history model as they are generated.

In this paper, we describe a prototype Memory Assistant tool we have developed that employs this approach with the goal of helping people with slight to moderate memory loss keep track of important objects placed around their home. Such memory difficulties are a significant issue in the aging population, and a major factor contributing to institutionalization of people who might, with modest human and machine assistance, be able to live on their own for a substantial additional period. The tool has been implemented in a model home that is part of the Center for Future Health at the University of Rochester, and demonstrated to work robustly with a small collection of ordinary 3D objects with multiple cameras surveying multiple rooms.

## 2. Previous Work

Tracking involves following a set of features through a sequence of images. Various different features have been used, such as contours [3], corners [20], shape and color [25]. Features detected in one image are matched to corresponding features in the previous image. Feature detection from one image to the next is frequently unreliable due to noise and occlusion. To handle this problem, researchers have employed techniques such as the Kalman filter [1] and conditional density propagation (Condensation) algorithm [3]. Both these algorithms work by estimating, or predicting, the location of the features being tracked in the next

image, and using the error between the predicted and measured location to update the predictive model.

Image-based object recognition techniques are more general and more easily trainable than those that employ geometric models. Currently, the most successful general object recognition systems are in this category. Most operate by finding the closest match of a target image representation to a large number of stored prototype representations of objects. Advantages of this approach include ease of training and being fairly general in object types that can be recognized.

One of the first examples of image-based object recognition was the use of neural nets to recognize wire objects [15]. Perhaps the best known work is that of subspace methods, used to recognize faces [23] and later more general objects [11]. Based on principal component analysis, these approaches reduce the image space to a subspace that captures most of the variation in the image data set. An object is represented as a manifold in the subspace, and recognition is done by projecting an unknown image into the subspace and locating the nearest manifold.

Subspace methods tend to be sensitive to clutter and occlusion. Techniques based on local image features address this problem. Several types of local image features have been explored. Examples include the response to a set of steerable filters [16], scale-invariant features [9], differential invariants [18], subspace approach applied to segmented regions [7], and context patches based on contours [19].

Our Memory Assistant has a lot in common with the field of video surveillance. Both are concerned with watching the movements of people and objects within a scene. Algorithms developed for the traditional surveillance domain may prove useful in our home health domain and vice-versa. Therefore, we present a short review of relevant work from the field of video surveillance.

Typical video surveillance techniques involve tracking humans. MIT's PFinder system tracks colored blobs corresponding to a user's head, hands, torso, and feet [25]. Rehg *et al.*, use motion, color, and stereo triangulation to find users in front of a smart kiosk [17]. The VSAM project detects and tracks multiple people and vehicles in cluttered scenes using multiple cameras [4]. Simple classification in human, vehicle, or clutter categories is performed.

Systems that detect simple activities involving interaction with tracked objects include Morris and Hogg [10] who statistically model the interactions between people and cars, and effectively quantify the statement, "Moving slowly close to a number of cars is unusual." The W<sup>4</sup> system detects and tracks multiple people, carried objects, and exchanges [6]. The TI system tracks objects and produces a graph representation of their spatio-temporal relations [5]. Events involving objects (appearance, disappearance, entrance, exit, deposit, removal, motion, and rest) are de-

tected by a rule-based system. Irani and Anandan describe a different approach to representing tracked movement in a scene [8]. The MIT AI Lab's forest of sensors project uses an adaptive background model to track and classify moving objects. It also learns typical patterns of activity for a site so unusual activity can be detected [21].

### 3. Memory Assistant Overview

The basic idea for the Memory Assistant is to have a set of pan, tilt, zoom cameras monitor specific locations in the home, such as the kitchen counter, bedroom dressers, etc., for movement of objects and agents, using a wide field of view. After local activity has ceased, the system attempts to identify compact regions of change that might correspond to objects added to or removed from the scene, or portions thereof. The pan, tilt, zoom capability is then used to take close-up, high-resolution images of regions where change has occurred, and the images are sent to an object recognizer for evaluation. This checks first for disappearance of objects previously recorded to be near the location of change, and then for the appearance of objects of interest. The recognizer employed is a feature-based system for rigid objects that is invariant to translation, scale, and 3D rotations, and robust to significant background clutter and modest occlusion. It employs shape-based features and thus is not dependent on distinctive color or stable lighting.

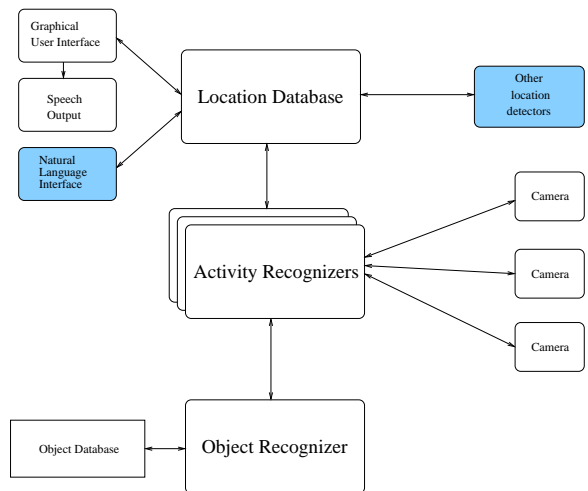


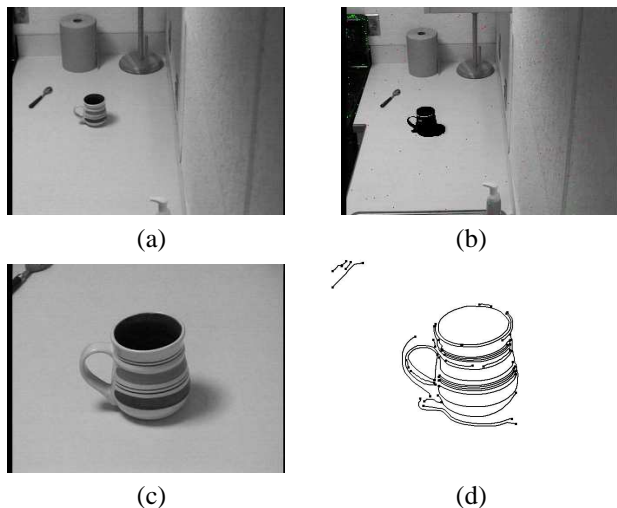
Figure 1. System Overview

An overview of the Memory Assistant system is given in Figure 1, with possible augmentations shaded. Each camera watches a designated location, such as the kitchen counter. An activity detector determines when there is activity (substantial movement) in the scene. After the activity ends, an object discovery system detects compact regions of the

scene that have changed. The camera then pans, tilts, and zooms to take a close-up image of each of these regions (up to some limit). These images are sent to the object recognizer for analysis. If an object is detected in one of these places, a found event for the object is generated and sent to the location database. If no object is found in a place where an object was previously located, then a lost event is generated for that object and sent to the location database.

The user can then query the location database for current and past locations of the objects. This is done via a user interface that employs a touch screen, visual cuing, and spoken and printed language.

Figure 2 shows the detection and recognition of a cup. (a) shows the cup on the counter. (b) shows the object discovery subsystem hypothesizing the cup as a foreground region. (c) shows the zoomed image of the cup used for recognition, (d) shows extracted curve features used by the recognition system.



**Figure 2. (a) The cup on the counter, (b) the cup detected by the activity recognizer, (c) zoomed image used for recognition, (d) curves used in recognition**

#### 4. Activity Detector

It is the responsibility of the activity detection system to watch the scene and detect changes that may correspond to activity in a scene or to the appearance or disappearance of an object. The overall approach employs a multiple-model per pixel architecture somewhat similar to that described in [21], but with various modifications. Basically, at each pixel, the system maintains one or more Gaussian models

(up to some small limit, generally 3 in our experiments) that represent the background, and any recently observed foreground values. Each model maintains a mean color value, and a variance estimate, and these values are adaptive with a time constant that can be adjusted to reflect variable conditions and frame rates. Each model also maintains history information in the form of the estimated percent of the time it was chosen to explain the pixel value on several time scales (e.g. 1, 10, 100, and 1000 seconds).

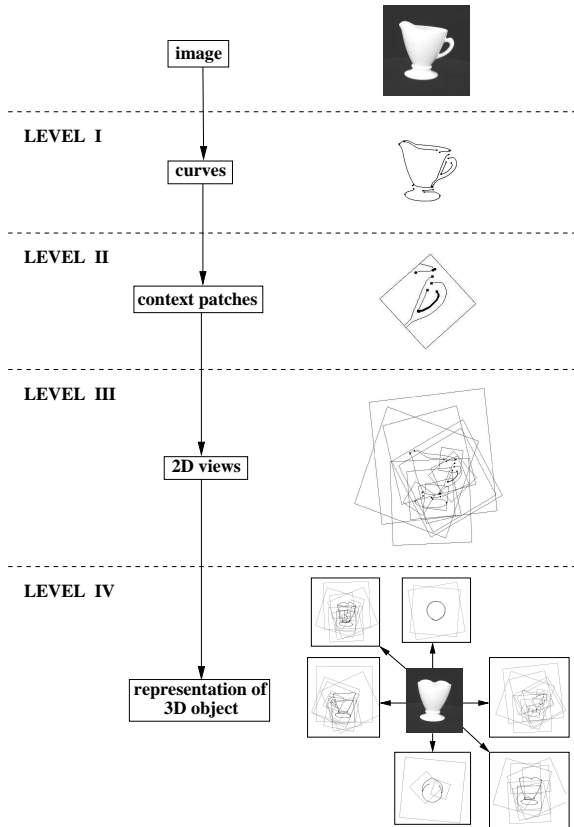
Each model is labeled as background or foreground, and the models are sorted in order of a “value” parameter, which is essentially an estimate of the value of the model in explaining recent history in minimum description length sense, where the bit cost of the model is added in amortized fashion to the bits required to encode the residual. This penalizes little-used models, and ones with high variance. Matching is not done by finding the best match, which leads to undesired model creep as objects come and go, but by a first match procedure, where models are compared in order of their value parameter, and the first model that matches the data within a given confidence region (e.g. 99 or 99.9 percent) is taken as the explanation. If no model matches, the one with lowest value parameter is kicked out, and a new foreground model is initialized.

The value of the multiple models is that movement and (multiple) new objects can be detected with high sensitivity, without losing the old background model, which is immediately detected once the new objects depart. To allow new objects to be eventually ignored, we permit persistent foreground labels to convert to background after some (generally long) time. The old background is generally still there as well.

Information is maintained for each pixel about which model was last active, and when the active model last changed. This, in conjunction with the history and label information stored with each model, permits a substantial amount of reasoning to be done. We start with a single background model at each pixel, acquired during a short initialization phase. We decide if activity is present by looking for recent changes in the matched model.

The system runs in both monochrome and rgb color modes. We have found the monochrome mode provides adequate detection in most of the situations we have tested. With various efficiency optimizations, the monochrome activity detection process runs at approximately 15 Hz on half-scale (320 x 240 pixel) images, on a 1.5GHz Xeon processor. The color version is about 30% slower.

To discover potential objects, we integrate activity over the entire image. If this exceeds a threshold, we conclude that something is going on, (e.g. a human agent is in the scene), and we then wait for activity to go below a threshold for some specified quiescent period. At this point, we perform a connected components analysis on the foreground



**Figure 3. Perceptual Grouping Hierarchy**

labeled pixels, and extract compact components in an appropriate size range for analysis. These regions represent potential objects, removed from or added to the scene.

For each such region (up to some limit), the cameras are instructed to pan, tilt and zoom so that the foreground region is approximately contained in the central half of the image. These images are sent to the object recognition system for analysis. Due to shadows, missed sections of the object, and inaccuracy in the kinematic model of the camera, the location of the region is imprecisely known in the high-resolution image, and hence a recognizer must be used that is robust to location, scale, clutter, and minor occlusion, as well as to 3D rotation and lighting. Since frame-rate response is not required, we can take several seconds to run the recognition process without negatively affecting performance. This allows fairly sophisticated algorithms to be used. We use a high-performance, 3D recognizer we have developed previously.

## 5. Object Recognizer

The object recognizer we use is feature-based and employs a perceptual grouping hierarchy, see Figure 3[12]. A

3D object is fourth-level feature represented by a group of 2D views. Each of these 2D views is third-level feature represented by a group of overlapping *local context patches*. Each local context patch is a second-level feature consisting of a windowed group of segmented curves (first-level features) that is centered on and normalized by a *key curve*. These first-level curve features are obtained from the image by low-level contour segmentation process. Currently, we use only connected contour segments as first-level keying features, but other primitives (e.g. regions or ribbons) could be similarly employed.

When provided with an image possibly containing one or more unknown objects, the recognition starts by generating context patches for every segmentable contour in the image (often several hundred). Each context patch is compared against a database of (several tens of thousands of) context patches extracted from training views, producing from 0 to a few hundred matches. Each match generates evidence for a hypothesis for a 2D view (third-level group) and pose parameters. Evidence for consistent hypotheses are merged, using a Bayesian evidence combination strategy yielding a total evidence score for each 2D view and pose. The 2D view and pose with the most evidence is considered the mostly likely object, and additional hypotheses (e.g. for other object classes) can also be extracted.

The evidence score can then be used to compute the probability of a false match. As part of the training phase, the mean and standard deviations of evidence scores for the images with the object present and with the object absent are calculated. The false match probability can be computed by the likelihood that the evidence score was generated by the object absent distribution. This is a true probability that allows us to set object detection thresholds in a principled fashion. Full details on the recognizer and evaluation are available in [13, 14].

## 6. Location Database

The location database stores, collates, and answers queries about object locations. A location detector can report the appearance or disappearance of an object to the database. The user interface or other programs can then query the database for a list of objects currently being tracked and their current and past locations.

The following is a comprehensive list of commands accepted by the location database.

- List tracked objects – List all objects currently being tracked by the database and the time of the last event concerning that object.
- Report Object Appearance Event – Used by detectors to report the appearance of an object from a scene.

- Report Object Disappearance Event – Used by detectors to report the disappearance of an object from a scene.
- Query Current Location – Return the most recent event associated with an object.
- Query Previous Location – Return previous events associated with an object. This option is useful to explore the history of a given object.
- Delete Event – Remove an event from the database. This option is useful when a detector misidentifies an object.

Currently, the only location detector in use is the vision-based object recognizer, but the location database is capable of storing location information from other sources such as infrared (IR) tags, radio frequency (RF) tags, or even other specialized vision algorithms (e.g. face detectors).

## 7. User Interface

The user interface was designed to be intuitive and easy to use. By utilizing a touch-screen interface, the need for a keyboard and mouse were eliminated. The user is presented with a set of images of the objects the system is tracking. To query the objects current location, the user simply presses the image of the object he wishes to locate (See Figure 4).

The system presents the object location to the user using a multi-modal approach. First, using the Festival speech synthesizer [22], it speaks the object's location. For example, "The cup is on the kitchen counter". This text is also displayed for the user to read and provides the user with the semantic location of the object. Second, the user is presented with a wide-angle image of the object's current location with the object outlined with a green box. This provides visual and spatial cues for the location of the object.

If the current location of the object is not known, the system informs the user that it does not know where the object is currently, but also provides the last location the object appeared in. The idea is that even though the system cannot tell the person exactly where the object is right now, it still might be helpful to know where it was last seen. This information may help trigger the user's memory as to what they did with the object.

In addition, the system provides facilities for viewing the history of object locations and deleting incorrect locations. Humans, even those with some memory loss, should be able to recognize when the system has made a mistake and correct it.

The current system consists of three machines – two dual 1.5GHz Xeon processor machines, and a 1GHz Pentium 4. The dual processor machines each handle the activity recognition for two Sony EVI-D30 pan, tilt, zoom cameras. The



Figure 4. User Interface Window

uniprocessor runs the location database, the voice synthesizer, and the user interface. The total system watches four surfaces in three separate rooms of a model smart house. For demonstration purposes, the system uses a three object database. However, the object recognition system has been tested in the lab with up to twenty-four objects.

## 8. Conclusion

There are real-world domains in which tracking objects using object recognition is a viable option. In a situation in which the objects of interest do not frequently move, one can watch for activity, and when the activity has ended, apply object recognition to find objects that may have moved and their new locations.

Our system is currently running in prototype smart home. It keeps track of objects placed in four locations – the bedside table, living room table, kitchen counter, and the bar area. The system performs well with a few people in the house moving the objects around. Performance degrades with crowds of people due to the fact that there is always activity being detected, so the system never gets a chance to take a close look at the regions that may have changed. Since the target users are elders living alone or with a care-giver, this situation should not occur too frequently.

## 9. Future Work

Future work will occur on several fronts. First, there is room for improvement of the underlying computer vision technology. Currently, the object recognizer is constrained to rigid objects. We are extending it to recognize less rigid objects, such as collections of keys. In addition, the activ-

ity recognizer will be improved to allow a single camera to watch multiple locations at the same time. Using background registration, the system will track objects in a larger area with the same hardware infrastructure.

Second, the system needs to be integrated with other smart home technologies. The most interesting of these is a spoken natural language interface that would allow the user to ask about object locations just as if they were asking another person. Infrared (IR) and radio frequency (RF) tags can also be used to track objects. In the current home, some objects and locations are equipped with IR tags. This information will be integrated with the current system.

Other issues include sharing of smart home resources with other applications. Other applications in the smart home include a gait recognizer, a skin mapper, and a medication advisor. Some utilize cameras similar to the ones used in the Memory Assistant. Building a infrastructure that allows these and other applications to share cameras, memory, and processors is being built. The goal is to have a system that is inexpensive and easily customizable to individual needs.

## References

- [1] A. Azarbayejani and A. Pentland. Recursive estimation of motion, structure, and focal length. In *IEEE Transactions on Pattern Analysis and Machine Intelligence*, volume 17, pages 562–575, 1995.
- [2] Y. Bar-Shalom and T. E. Fortmann. *Tracking and Data Association*. Academic Press, 1988.
- [3] A. Blake and M. Isard. *Active Contours*. Springer-Verlag, 1998.
- [4] R. Collins, A. Lipton, and T. Kanade. A system for video surveillance and monitoring. In *Proc. American Nuclear Society (ANS) Eighth International Topical Meeting on Robotics and Remote Systems*, 1999.
- [5] J. D. Courtney. Automatic video indexing via object motion analysis. *Pattern Recognition*, 30(4):607–625, 1997.
- [6] I. Haritaoglu, D. Harwood, and L. S. Davis.  $W^4$ : Real-time surveillance of people and their activities. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8):809–830, 2000.
- [7] C.-Y. Huang and O. I. Camps. Object recognition using appearance-based parts and relations. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 877–883, 1997.
- [8] M. Irani and P. Anandan. Video indexing based on mosaic representation. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 86(5):905–921, May 1998.
- [9] D. G. Lowe. Object recognition from local scale-invariant features. In *Proceedings of the International Conference on Computer Vision*, pages 1150–1157, 1999.
- [10] R. J. Morris and D. C. Hogg. Statistical models of object interaction. *International Journal of Computer Vision: IJCV*, 37(2):209–215, 2000.
- [11] H. Murase and S. Nayar. Visual learning and recognition of 3-d objects from appearance. *International Journal of Computer Vision*, 14(1):5–24, 1995.
- [12] R. C. Nelson and A. Selinger. A cubist approach to object recognition. In *International Conference on Computer Vision (ICCV98)*, January 1998.
- [13] R. C. Nelson and A. Selinger. Large-scale tests of a keyed, appearance-based 3-d object recognition system. *Vision Research*, 38(15-16), August 1998.
- [14] R. C. Nelson and A. Selinger. Learning 3d recognition models for general objects from unlabeled imagery: An experiment in intelligent brute force. In *International Conference on Pattern Recognition*, September 2000.
- [15] T. Poggio and S. Edelman. A network that learns to recognize three-dimensional objects. *Nature*, 1990.
- [16] R. Rao. Dynamic appearance-based recognition. In *Proc. of Computer Vision and Pattern Recognition*, 1997.
- [17] J. M. Rehg, M. Loughlin, and K. Waters. Vision for a smart kiosk. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition CVPR97*, pages 690–696, San Juan, Puerto Rico, June 17–19 1997. IEEE Computer Society Press.
- [18] C. Schmid and R. Mohr. Combining greyvalue invariants with local constraints for object recognition. In *Proceedings of Computer Vision and Pattern Recognition*, pages 872–877, 1996.
- [19] A. Selinger and R. C. Nelson. A perceptual grouping hierarchy for appearance-based 3d object recognition. *Computer Vision and Image Understanding*, 76(1):83–92, October 1999.
- [20] J. Shi and C. Tomasi. Good features to track. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 593–600, 1994.
- [21] C. Stauffer and W. E. Grimson. Learning patterns of activity using real-time tracking. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8):747–757, August 2000.
- [22] P. A. Taylor, A. Black, and R. Caley. The architecture of the festival speech synthesis system. In *The Third ESCA Workshop in Speech Synthesis*, pages 147–151, 1998.
- [23] M. Turk and A. Pentland. Face recognition using eigenfaces. *Journal of Cognitive Neuroscience*, 1991.
- [24] G. Verghese, K. L. Gale, and C. R. Dyer. Real-time motion tracking of three-dimensional objects. In *Proc. IEEE Robotics Automat. Conf.*, pages 1998–2003, 1990.
- [25] C. Wren, A. Azarbayejani, T. Darrell, and A. Pentland. Pfnder: Real-time tracking of the human body. Technical Report 353, MIT Media Lab, 1997.