Temporal Dynamical Interactions between Multiple Layers of Local Image Features for Event Detection in Video Sequences

Daniel Kottow*, Mario Köppen and Javier Ruiz-del-Solar
daniel.kottow@ipk.fhg.de|mario.keoppen.ipk.fhg.de|jruizd@cec.uchile.cl

Fraunhofer IPK, Dept. Pattern Recognition, Pascalstr. 8-9, 10587 Berlin, Germany
DIE, U. de Chile, Tupper 2007, Santiago, Chile

Abstract. In this paper an approach for storing and employing local image features in video processing is presented. The approach is based on the usage of memory cells representing local image features and (non-fixed) spatial positions, which are organized in memory layers. By assigning frame-based recall function and learning procedure to the cells, the memory layers establish a content-based auto-associative memory. Thus, they can be applied to solve several event detection tasks, as it is exemplified by dynamic background supression in a traffic scene, and counting of persons halting before a shopping window in an indoor scene. The case studies suggest that information gathered from the cells (like cell history based scoring values) can be used in various manners for video processing tasks circumventing the need for object segmentation and tracking, typical in many conventional background-differencing methods.

1 Introduction

Local image features are a common starting point for image analysis and typically computed at each image pixel: a new (vector) image representing the feature values is obtained. We propose an alternative way of storing and computing local image features called content-based memory layers. A memory layer stores features in a spatially organized way but allows access to the information only through local operations for recalling or learning a certain image position. These operations are based on a definition of matching between an image feature and features stored in the memory layer that allows user-defined tolerances for the feature value and its position. Section 2 defines content-based memory layers and their usage and illustrates particular system properties.

Event detection in video sequences refers to the analysis of video streams searching for predefined, anthropocentric events that may manifest diversely on the actual images. Applying all stages of traditional image processing to video sequences typically leads to algorithms that segment and track multiple moving objects; a task that gets exponentially more difficult as the number of

* This work was partially funded by a DAAD doctoral grant

J. Bigun and T. Gustavsson (Eds.): SCIA 2003, LNCS 2749, pp. 223–231, 2003.

objects in the scene increases. Event detection many times is built on top of these algorithms, as for example in [3]. Our work focuses on events that may be detected using motion information only, circumventing the need for individual object segmentation. In section 3 we present two algorithms for event detection in real-world environments using multiple memory layers and suitable temporal dynamics between them.

2 Content-based Memory Layers

Content-based memory layers store local image features in a spatially organized way. Thereto each feature held in a memory layer bears an associated image coordinate. Interactions between an image and a memory layer are localized spatially: first a local feature is extracted from the image at a certain position and then the memory layer is searched for similar features with an associated position close to the query position. When operating in a learning mode, local image features that are not recalled by the memory are added to it.

Section 2.1 defines a memory layer, the fundamental operations between a layer and an image; section 2.2 builds upon these definitions and introduces color features in this context. Section 2.3 illustrates some system properties processing images taken from a traffic scene [1].

2.1 Basic Definitions

We start by defining the basic element of our content-based memory called **a cell**. A cell is made out of a local image feature and an image coordinate $\mathbf{x} = (x, y)$ and may be written $c = \{f_c, \mathbf{x_c}\}$. In this context it suffices to say that a local image feature is the output of a feature extraction function $f(\mathbf{x}, I)$ that only depends on the values of the image I at the position $\mathbf{x}$ and a local neighborhood (features are typically vectors in $\Re^n$).

We define a predicate called **matching** between a cell $c = \{f_c, \mathbf{x_c}\}$ and an image I at a position $\mathbf{x}$ as:

$$m(c, \mathbf{x}, I; \varepsilon, \delta, f(\mathbf{x}, I)) = d(f_c, f(\mathbf{x}, I)) < \varepsilon \wedge d(\mathbf{x_c}, \mathbf{x}) < \delta \tag{1}$$

where $d(\mathbf{x}_1, \mathbf{x}_2)$ is the Euclidean distance in the image coordinate plane and $d(f_1, f_2)$ is a distance measure in the feature space to which both, f_c and $f(\mathbf{x}, I)$, belong. Matching introduces the fundamental operation of our system: it compares a stored bit of information $c = \{f_c, \mathbf{x_c}\}$ and a local image feature $f(\mathbf{x}, I)$ allowing control over the accuracy in the representation of feature values and positions through the parameters ε, δ respectively.

A **memory layer** is a collection of cells $C = \{c_i\}$, all bearing feature vectors in the same space, and a corresponding matching function. We define two basic operations on a memory layer. A **recall function** $R(\mathbf{x}, I, C)$ searches for a cell in C that matches the image I at the position $\mathbf{x}$:

$$R(\mathbf{x}, I, C) = \begin{cases} c_{match} & if\, \exists c \in C\, (m(c, \mathbf{x}, I; f, \varepsilon, \delta)) \\ \emptyset & otherwise \end{cases} \tag{2}$$

where c_{match} is any cell in C that satisfies the matching predicate (1). A **learning procedure** $L(\mathbf{x}, I, C)$ creates a new cell if the recall function fails:

$$L(\mathbf{x}, I, C) = \begin{cases} C \leftarrow C \cup \{c_{new}\} & if\, R(\mathbf{x}, I, C) = \emptyset \\ nothing & otherwise \end{cases} \tag{3}$$

where c_{new} is a new cell that must satisfy the matching predicate. The most straightforward way to generate c_{new} is by extracting a feature at the position where the recall function failed and setting the cell's feature and position accordingly.

An alternative way of recalling starts with a certain cell and determines a possible matching position. We define a **recall-cell function** as:

$$RC(c, I) = \begin{cases} \mathbf{x}_{match} & if\, \exists \mathbf{x}\, (m(c, \mathbf{x}, I; f, \varepsilon, \delta)) \\ \emptyset & otherwise \end{cases} \tag{4}$$

where $\mathbf{x}_{match}$ is any position where the matching predicate holds.

We define the **score of a cell** s_c as a dynamic variable, which is updated every time the recall-cell function is called, according to:

$$US(c; \lambda) = \begin{cases} s_c \leftarrow (1-\lambda)s_c + \lambda & if\, RC(c, I) \neq \emptyset \\ s_c \leftarrow (1-\lambda)s_c - \lambda & otherwise \end{cases} \tag{5}$$

where $\lambda \in [0, 1]$ is a user-defined adaptation rate. Initially, when creating a cell we set $s_c = 0$. Similarly, we define the update of a cell's feature value:

$$UF(c; \mu) = \begin{cases} f_c \leftarrow (1-\mu)f_c + \mu f(\mathbf{x}_c, I) & if\, RC(c, I) \neq \emptyset \\ nothing & otherwise \end{cases} \tag{6}$$

with $\mu \in [0, 1]$.

2.2 Further Definitions

Equations 2 and 3 process a single image position. A straightforward generalization for processing an image area is obtained by processing all positions in the area $A = \{\mathbf{x}_i\}$ independently:

$$R(A, I, C) = \{R(\mathbf{x}_i, I, C)\}_{\mathbf{x}_i \in A} \tag{7}$$

where $R(\mathbf{x}, I, C)$ may be replaced by the learning function. Note, however, that learning is dependent on the order in which the positions are processed, because each learning procedure may add a cell to C which in turn influences subsequent learning results.

Seed growing strategies consider areas of similar image values which by definition are of previously unknown spatial extent. Given a set of seeds $S = \{\mathbf{x}_i\}$ we define a seed growing algorithm for learning in Algorithm 1.

Until now we have not needed to specify a definite kind of local image feature for the operations with a memory layer. Now we introduce **color cells** storing local color information. Given a (possibly multi-channeled) image I we define a color feature as the mean of the image pixel values contained in a square of dimension l centered at $\mathbf{x}$:

$$f_{mean}(\mathbf{x}; l) = \sum_{\mathbf{dx} \in d\Omega} I(\mathbf{x} + \mathbf{dx}) \tag{8}$$

where $d\Omega = \{\mathbf{dx} \mid |\mathbf{dx}| < l\}$. Note that a color feature will have the same dimensionality as the image pixel values. We are currently working with RGB color images and use the Euclidean norm to measure the distance between mean color features.

2.3 System Properties

Consider the images shown in fig. 1. In this section we will learn and recall these images using a memory layer of color cells demonstrating the usage and some properties of the system. Cells resulting from learning a whole image area using eq. 3 are shown in fig. 2.

Content-Based Auto-Associative Memory. Once an image area has been learned, the system knows about position and mean color for every cell. A great amount of information reduction has taken place and sensible details seem lost, as fig. 2 illustrates. The original image used for learning cannot be reconstructed using only this information. But if we use eq. 2 to recall the same scene taken some seconds later (shown in fig. 1 below) we can make out even small differences between both images (as shown in fig. 3 above). Having a precise notion of information that is not arbitrarily retrievable is an essential characteristic of content-based memories.

User and Data Driven Ressource Usage. The user-defined parameters ε, δ define the accuracy of the feature value and locus representation, respectively, thus giving the user control over the vigilance level of the content-based memory. The dual control over the accuracy of the feature representation results in an uneven cell density depending on the complexity of local features in a scene as it is readily observed in fig. 2. When learning an image sequence, several cells bearing diffferent feature values may accumulate even at a single position if there is a periodically changing image texture, as e.g. specularities in the water.

Algorithm Complexity. Recalling or learning any image position needs to verify the matching condition given by eq. 1 iteratively for a whole set of cells C. Using simple caching and a hashing technique for two-dimensional planes called spiral nearest neighbor described by Bentley and Weide [4] we could reduce the computations required for recalling or learning an image position to a single feature extraction and the comparison of in average 2-5 cell features. When searching for novelties in an image area we can greatly reduce computation to a small fraction of the image positions contained in the area by sampling the area and employing the seed growing strategy presented in Algorithm 1.

3 Event Detection in Video Sequences using Multiple Memory Layers

Until now we have been using a single memory layer. In this section we will use multiple memory layers and suitable temporal dynamics between their cells in order to do motion segmentation in video sequences. In the first video showing a traffic scene [1] we segment foreground objects from background information making the background include automatically halted objects and expell them when they start moving again. In the second video showing people in front of a shopping window [2] we discriminate the persons looking into the window.

3.1 Dynamic Background Suppression

In this section we define an algorithm for the segmentation of moving objects using a dynamically updated background memory layer. The updates occur in a twofold way: color features representing the background are adapted in a slow, linear manner; and halted objects are included in the background using a locally precise learning procedure induced by foreground features that remain active during a long time. Background cells are expelled once their features become obsolete.

First we define a **dual learning procedure** for two layers C_{bg}, C_{fg} bearing background and foreground cells, respectively:

$$L_2(\mathbf{x}, I, C_{bg}, C_{fg}) = \begin{cases} L(\mathbf{x}, I, C_{fg}) & if\ R(\mathbf{x}, I, C_{bg}) = \emptyset \\ nothing & otherwise \end{cases} \tag{9}$$

This procedure allows the creation of foreground cells guaranteeing that they represent local features that are not present in the background layer.

The algorithm initializes a background memory layer of color cells by learning all positions from an initial frame. In subsequent frames we first apply the seed growing strategy using (9). Then we apply the recall-cell function to all background cells and adapt its score and feature according to the delta rules given in (5) and (6). If the score of a background cell drops below a threshold, the cell gets removed. This cleans our memory from obsolete features, as e.g. removed background objects. We also apply the recall-cell function to all foreground cells and update their score accordingly. After the lifetime of a foreground cell exceeds a certain number of frames it is removed. If one of them has a particularly high score, it induces learning in the foreground. This implements the idea that a foreground cell scoring for a large number of frames belongs to a halted object and should be represented in the background. The pseudo-code for dynamic background suppression is given in Algorithm 2.

In fig. 3 we show how these cell dynamics influence the segmentation results. The most important difference is the absorbtion of static objects by the background; we see the streetcar ghost has disappeared as well as the person standing at the lower left street corner. Uniform little recall failures are eliminated by the slow adaptation of the background cell features. In fig. 4 we show how the number of cells in each memory layer evolves in time. The events on the video are

reflected in these plots. In general, moving objects that enter the scene induce an increase in the number of foreground cells and vice-versa. The fluctuation in the number of background cells, which captures a red-light-event given by a queue of halted cars, is a direct consequence of the storing properties of memory layers as described in section 2.3 and reflects the fact that cars are more complex in term of color features than an empty street.

3.2 Discrimination of Moving Foreground Objects

The PETS 2002 workshop provided some video sequences showing people walking and standing in front of a shopping window and asked participants to implement algorithms counting the persons passing by and discriminating those looking at the shopping window. In the following we present an algorithm which only processes and counts persons standing in front of the window. It does not track or segment people; it rather discriminates halted foreground blobs from moving ones (see fig. 5) and integrates their areas in order to estimate the number of people in front of the window. The algorithm uses three memory layers for this purpose, a background layer, a layer for moving objects and another for the halted ones.

We define a **triple learning procedure** for three layers $\zeta = \{C_{bg}, C_{so}, C_{do}\}$ bearing background, halted (static) object and moving (dynamic) object cells, respectively:

$$L_3(\mathbf{x}, I, \zeta) = \begin{cases} L(\mathbf{x}, I, C_{do}) & if\ R(\mathbf{x}, I, C_{bg}) \wedge R(\mathbf{x}, I, C_{so}) = \emptyset \\ nothing & otherwise \end{cases} \tag{10}$$

This procedure allows the creation of cells for moving objects guaranteeing that they are neither present in the background layer nor in the layer for halted objects.

This algorithm (Algorithm 3) is very similar to dynamic background suppression being the main difference the inclusion of the third layer for halted objects. C_{so} takes on the role of C_{bg} in the Algorithm 2, while in this case the background is not updated at all. Another major difference is the inclusion of an additional criterion for removing cells from the halted objects layer; if a cell has only a few neighboring cells it is supposed to belong to a difference in the background image values due to varying lighting conditions rather than a halted person and is consequently removed. This condition is checked only after the cell has reached some maturity, which allows gradual growing of static areas.

We now describe how to estimate the number of halted persons in a frame given the cells in C_{so}. First we compute the number of static object cells $nc(x)$ contained in a rectangle of width w, centered at the horizontal coordinate x and extending over the vertical coordinate y up to a height h:

$$nc(x) = \sum_{c \in C_{so}} \sum_{y<h} \sum_{|dx|<w} \delta(x + dx - x_c)\delta(y - y_c) \tag{11}$$

where $\delta(x)$ is a discrete (kronecker) delta function. To obtain the number of halted persons in each frame, we threshold $nc(x)$ and count the number of

nonzero values which are sufficiently far apart in order to belong to different persons. Once $nc(x)$ is computed for the whole sequence, smoothing along the time axis is done using a median filter. The results are shown in fig. 6. They compare very favorable to a subjective ground truth, being the number of estimated people standing in front of the window correct in ~95% of the video. Since the inclusion and the removal of cells is subject to time constants, static cells survive if they are temporarily occluded by a moving object passing in front of them, which enhances the robustness of the results. Errors occur when people stand with their back to the shopping window or slowly walk away along the y-axis.

4 Conclusions

This work has touched upon two aspects concerning the processing of video sequences. We introduced a novel way of storing local image features, called memory layers, and defined operations for recalling and acquiring local image features. We show that memory layers have some interesting properties when compared to the storage of local features in a fixed coordinate system: non-uniform spatial feature density depending on the complexity of the scene, user-defined parameters for value and position accuracy of the feature representation and content-memory like behaviour.

In the second part of this work we defined algorithms using multiple memory layers for event detection in video processing. We focused on events which may be detected using motion information only, no shape analysis or object tracking is performed. A significant advantage of such an approach is the ability to directly focus on global properties that are easy to compute and avoid an exponential increase in complexity due to many objects in the scene. Naturally, such an approach restricts the class of events that can be detected with such systems. We are also working on algorithms using memory layers that allow multiple objects to be tracked using edge-based local image features instead of color.

References

1. Institut für Algorithmen und Kognitive Systeme Kognitive Systeme - Image Sequence Server. `http://i21www.ira.uka.de/image_sequences`.
2. Third IEEE International Workshop on Performance Evaluation of Tracking and Surveillance. `http://pets2002.visualsurveillance.org`.
3. S. Richetto J.H. Piater and J.L. Crowley. Event-based activity analysis in live video using a generic object tracker. In *Third IEEE Int. Workshop on Performance Evaluation of Tracking and Surveillance*, 2002.
4. B.W. Weide J.L. Bentley and A.C. Yao. Optimal expected-time algorithms for closest point problems. *ACM Transactions on Mathematical Software*, 6(4):563–580, 1980.

Appendix 1: Figures

Figure 1: Frames 0000 (above) and 0400 (below) from one of the traffic videos available at [1]. All frames are gray-scale images of size 768x400.

Figure 2: Color cells of size 3x3 representing a cutout of frame 0000 shown in fig. 1.

Figure 3: The upper image shows a recall of frame 0400 using background cells learned from frame 0000. The lower image is the resulting motion segmentation at frame 0400 achieved with the Algorithm 2.

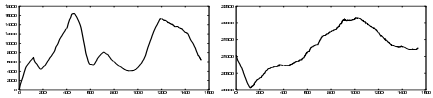

Figure 4: The time evolution of the number of cells in the foreground (left) and background (right) layer for the traffic scene video [1].

Figure 5: Frame 0569 from the third testing video sequence of the PETS2002 dataset [2] (above). Below its representation using cells computed by Algorithm 3. Light grey is used for cells from the background layer, darker grey for the dynamic and black for the static object layer.

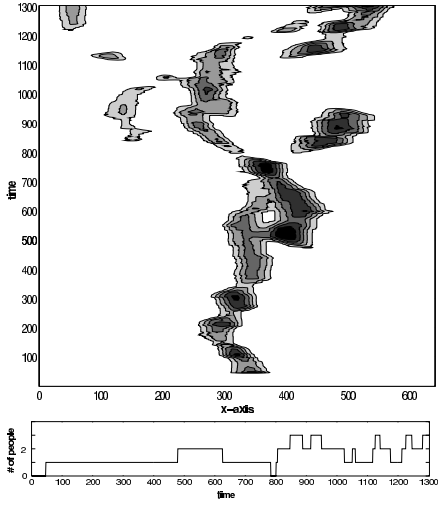

Figure 6: Contour plot of $nc(x)$ (above) and estimated number of people standing in front of the shopping window (below).

Appendix 2: Algorithms

Algorithm 1: Learning by Seed Growing

```
LearnBySeedGrowing(S):
  while S ≠ ∅:
    x ← PopItem(S)
    if NotProcessed(x):
      L(x, I, C)
      if NewCell(C):
        foreach x_n ∈ N(x):
          PushItem(S, x_n)
endLearnBySeedGrowing
```

$S = \{x_i\}$ is an ordered set of positions. `PopItem` removes and returns the first position in S. `PushItem` adds a position to the end of S. `NotProcessed` returns false for any position already processed. `NewCell` returns true if the last learning procedure added a cell to C. $N(\mathbf{x})$ returns the 4-connected neighbors of $\mathbf{x}$.

Algorithm 2: Dynamic Background Suppresion.

```
Initialize(I_0, A):
  L(A, I_0, C_bg)
endInitialize

ProcessFrame(I, A):
  S ← SampleSeeds(A)
  Learn2BySeedGrowing(S, C_bg, C_fg)
  foreach c ∈ L_bg:
    RC(c, I, C_bg); US(c; λ); UF(c; λ)
    if s_c < s_min:
      RemoveCell(c, C_bg)
  foreach c ∈ C_fg:
    RC(c, I, C_fg); US(c; λ); a_c ← a_c+1
    if a_c > a_kill:
      RemoveCell(c, C_fg)
      if s_c > s_high:
        foreach x ∈ dΩ(x_c, r):
          L(x, I, C_bg)
endProcessFrame
```

s_{min} is the minimal score for background cells to stay alive. a_c is the number of frames a foreground cell has been in alive; when it reaches a_{kill} the cell is removed. If a cell's score reaches s_{high} it induces learning in the background layer. $d\Omega(\mathbf{x}_0, r)$ is a function returning all positions that satisfy $|\mathbf{x} - \mathbf{x}_0| < r$.

`SampleSeeds` is a function that evenly samples the area A. `Learn2BySeedGrowing` is the Algorithm 1 using eq. 9 as the growing criterion. `RemoveCell` removes a cell from its memory layer. All other variables are defined in the text.

Algorithm 3: Discrimination of Static Foreground Objects

```
Initialize(I_0, A):
  L(A, I_0, C_bg)
endInitialize

ProcessFrame(I, A):
  S ← SampleSeeds(A)
  Learn3BySeedGrowing(S, C_bg, C_so, C_do)
  foreach c ∈ L_so:
    RC(c, I, C_so); US(c; λ); a_c ← a_c+1
    if (s_c < s_min) ∨
       (a_c > a_min ∧ |dΩ(c, r, C)| < n_min):
      RemoveCell(c, C_so)
  foreach c ∈ C_do:
    RC(c, I, C_do); US(c; λ); a_c ← a_c+1
    if a_c > a_kill:
      RemoveCell(c, C_do)
      if s_c > s_high:
        foreach x ∈ dΩ(x_c, r):
          L(x, I, C_so)
  CountHaltedPeople(C_so)
endProcessFrame
```

`Learn3BySeedGrowing` is the Algorithm 1 using eq. 10 as the growing criterion. $d\Omega(c, r, C)$ returns all nearby cells $\{c_i\}$ in C whose positions satisfy $|\mathbf{x}_i - \mathbf{x}_c| < r$. if a cell has less than n_{min} nearby cells and its age is greater than a_{min} the cell is removed. `CountHaltedPeople` estimates the number of people standing in front of the window and is described in the text. All other variables and procedures are the same as in Algorithm 2.