International Journal of Image and Graphics
© World Scientific Publishing Company

A User Interface Design for Acquiring Statistics from Video

Man-Kang Leung and Chi-Wing Fu
The Hong Kong University of Science and Technology
Hong Kong
{*cskang,cwfu*}*@cse.ust.hk*

Received June 12, 2007
Revised November 2, 2007
Accepted November 8, 2007

This paper introduces a graphical user interface approach to facilitate an efficient and timely generation of statistic data from input videos. By means of a carefully-designed graphical user interface, users can interactively add in various kinds of markers, known as the *statistic inducers*, on the screen of an input video to specify the areas of interest corresponding to the locations of relevant events. These inducers are in the form of two-dimensional points, lines, polygons, and grids, and can be put on the video screen with great ease. Using these inducers, we not only can efficiently customize the system for a given statistic generation task; in addition, we can also precisely constrain the time-consuming space-time video analysis process (as well as any additional analysis process like optical flow computation or object recognition) on the user-specified areas. To demonstrate the efficacy of the proposed approach, we developed a prototypic system and experimented it in two different statistic generation cases: dormitory light switching and road traffic. In both cases, we just need a few minutes of UI customization time to set up the inducers; once this is done, timely statistics can be automatically generated subsequently.

Keywords: User interface, video analysis, statistics, change detection

1. Introduction

Video is a comprehensive multimedia element widely used in many applications and electronics. With recent advances in video capture, storage, and transmission technology, and also with the dropping in the cost of video capturing device, the use of videos is no longer limited to entertainment industries for film making or television production; we now have a wide variety of commodity video-based applications like tele-conferencing, home surveillance, security cameras, and instant news broadcasting on mobile devices.

Though videos are so popular in everyday life, most conventional video applications still regard videos as stacks of consecutive images only; video is merely regarded as a data storage for capture, transmission, and display. Rather, the application, or the interpretation, of videos is not limited to this; we can interpret the contents stored in a video, as a three-dimensional spatio-temporal data volume documenting both spatial and temporal events (or activities) happened, or being happened, in a physical environment continuously over a well-defined period of

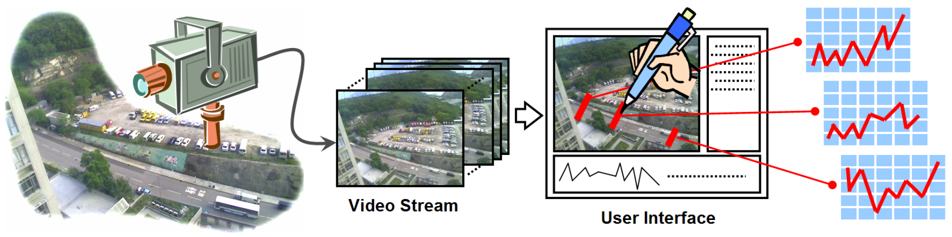


Fig. 1. Overview of the "Statistics from Video" framework.

time. From this point of view, we can further bring in high-level video applications with the help of ingenious image understanding and computer vision techniques, see Subsection 1.1 for some interesting examples.

Why "Statistics from video"?

Statistic data is a highly valuable resource widely employed in many disciplines nowadays; from experiments in scientific studies to decision-making in commercial enterprises, statistic is an objective mean not distorted by personal bias or emotion, and is based entirely upon observations. However, the situation we commonly have with statistics is that the effort required to get raw statistic data is usually highly enormous both in terms of manpower and time.

Regarding this, what this paper focus on is the application of videos in generating statistic data. Since a video documents both spatial and temporal events, if we can employ it as a mean to generate raw statistic data (through video analysis), we not only can greatly save the manpower and time involved in compiling the statistic data, but can also significantly shorten the time in obtaining the statistic data as well. Timely statistic data could be immediately available right after the happening of related events.

What is "Statistics from video"?

The central idea in the proposed "Statistics from video" framework, see Figure 1, is a graphical user interface design to link the two entities, video and statistics, together; here, we can capture events through a video, or directly from a live video stream, and efficiently generate the related statistic data (in the form of time-series data) by means of some video analysis methods. The user interface is the middleman for users to efficiently input prior semantic information to customize the system for a given statistic generation task.

Using this user interface approach, we can tremendously reduce the manpower involved, and also significantly speedup the statistic generation cycle, which, in turn, can further facilitate the design of high-level spatio-temporal queries or analysis in the domain of data mining and information retrieval. In details, to bring this idea into practice, we introduce two important elements in the proposed approach:

- The first element is a *specially-designed user interface* for users to efficiently

mark up regions of interest with respect to relevant events; here we propose the *statistic inducer*, as an efficient media for users to do the customization.
- The second element is the *attachment of callback functions*, so that the statistic generation process can be specialized for different statistic generation tasks.

Using this framework, we can efficiently and almost automatically (it only takes users a few minutes' time to set up the inducers) obtain statistic data, such as 1) the traffic condition (e.g., statistics about the number of cars through a certain location per minute), 2) the sleeping habit in a dormitory, and 3) crowd control or rate of people flow in a mall, etc. The overview of the framework and the user interface design are presented in Section 2; the video analysis component is presented in Section 3, while the implementation details and two statistic generation examples are reported in Section 4.

1.1. *Related work*

Computer Graphics and Images In recent research of computer graphics, there are emergent topics like video-based rendering, and computational photography and video. In line with the main idea in this paper, a graphical user interface (GUI) is designed and provided to the users as an interactive visual computing environment that takes in user interaction to process and edit video/image data. One example is the work of Bhat et al.[5], where users can employ an interactive GUI to edit videos of water flows and synthesize novel water flow animations. Agarwala et al.[4] tracked contours in a video sequence for rotoscoping and cartoon animation creation. Sand and Teller[33] proposed an interesting and robust video matching idea, which innovated some novel video-based applications as well as the development of various special video effects. The GUI system designed by Wang et al.[45] can transform videos into cartoon animations in a non-photorealistic style. In addition, Wang et al.[43] also developed an interactive video segmentation system for cut-and-paste of video-based objects among different video sequences. A similar UI system was reported by Li et al.[20]. Hengel et al.[42] designed a GUI for users to create 3D models out of videos. Other than using interactive GUIs to process videos, interactive GUIs are also designed for applications like: manga colorization[27], a photo clip art system[18] for inserting image-based objects into existing photographs, and a sketch-based UI system[47] to specify 3D shapes through normal transfer.

Visualization Pingali et al.[25] applied real-time video analysis technique to extract the motion trajectories of tennis players and the ball, so that we can visualize the actions taken in the form of a 3D virtual replay, and look at the ball trajectory at any angle. Daniel and Chen[10] considered a video as a 3D volume data and applied volume rendering to visualize the events and activities happened in the captured video. Román et al.[30] made use of a sideways-looking video captured from a moving vehicle to create a multiperspective view of the scene; this result facilitates the visualization of urban landscape.

Multimedia In the area of multimedia research, there are many different kinds of video-based applications; Cavallaro et al.[7] developed an automatic algorithm for tracking multiple objects in a video sequence; interactions between both low and high level image analysis results were employed. Ou et al.[24] built the DOVE (Drawing Over Video Environment) collaboration system; users can draw on videos to share ideas over a video feed. Wu et al.[48] detected text on road signs by analyzing videos captured on a moving vehicle. Xu et al.[49] developed an automatic music video summarization system by simultaneously looking at the extracted chorus from the music track, and the extracted lyrics and shots from the video track at the same time. He and Zhang[16] invented an interesting vision-based multimedia application called the Real-Time Whiteboard Capture System (RTWCS); the video stream captured from a low-cost video camera can be analyzed in real-time to produce line strokes and drawings, lively shared in a tele-conference environment.

Change Detection Apart from presenting related work on video rendering and analysis, we would also like to discuss some related work on change detection as change detection is a major component inside our *statistics from video* framework.

Basically, Change detection has a long history in the literature of image processing and computer vision. Radke et al.[28] presented a systematic survey on change detection, and here we briefly review some related techniques in this area:

- One straightforward approach to detect changes in a video is by computing the difference between successive frames. The *simple differencing* method[31,32] thresholds the absolute difference on individual pixels between consecutive frames and looks for changes. Techniques closely related to this idea include the *linear dependence change detector (LDD)*[11], the *change vector analysis (CVA)* technique[22], *image ratioing*[34], and *simple difference on subsampled gradient images*[36].
- Instead of comparing consecutive images only, we can track intensities of pixels over time (either pixel-based or block-based), model the background (and maybe the foreground also), and apply statistical hypothesis to detect changes[3,2] when a new image is given. This approach is in general called *background modeling*; several modeling methods have been proposed: *a single Gaussian model*[3,46], *a mixture-of-Gaussian model*[35,26], a predictive model like the *Wiener filter*[40], the *Kalman filter*[17], *the non-parametric kernel density estimation*[12], *the hidden Markov model (HMM)*[29,37], *the Markov random field*[2], and the *minimum description length (MDL) approach*[19].
- More than analyzing a given video pixel by pixel, we can also take into account the spatial domain of the video (and maybe the temporal domain as well), extract the structural entities such as edges, objects, or regions, and detect changes spatially. Related techniques include the *morphological filter*[38], segmentation-based methods[36] (such as mean shift segmentation[44,9]), detection using color edges[6], and the Wallflower hierar-

chical analysis system[40].

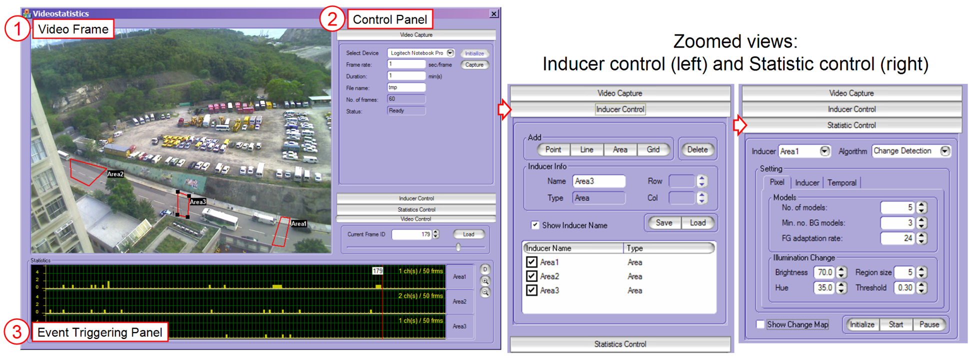
- Another stream of research in change detection is to handle the illumination changes in video so that the change detection process is illumination independent. This is especially significant when we consider “time of day” and “shadow” issues in the change detection. Conventional techniques dealing with this issue include *intensity normalization*[21,41], *homomorphic filtering*[39,1] for Lambertian surfaces, the *generic linear model*[23] for modeling the radiometric variation, the *principal component analysis (PCA)* technique[14] for illumination decomposition, and the *linear dependence change detector (LDD)*[11].

Further than that, some researchers combined several techniques above together in order to implement practical surveillance systems. Haritaoglu et al.[15] combined techniques like thresholding, the morphological filter, etc. in their W^4 surveillance system, whereas Collins et al.[8] also combined a series of change detection methods in their autonomous video surveillance and monitoring system. In this paper, we also employ several change detection techniques inside our *statistics from video* framework, but since our framework is a high-level design, it is independent of the change detection algorithms chosen.

2. The User Interface: Statistics from Video

2.1. *The User Interface*

Figure 2 presents the user interface we implemented in the prototypic system. To generate statistic data given an input video, the very first step we have to do is to mark up some regions of interest on the video screen corresponding to the relevant activities as to the statistic data to be inferred. In the example shown in the figure, some *statistic inducers* are marked on the road to watch over the cars crossing them over time; here, raw statistic data corresponding to this action can be generated accordingly, and later be compiled.



Select Device: Logitech Notebook Pro
Frame rate: 1 sec/frame
Duration: 1 min(s)
File name: tmp
No. of frames: 60
Status: Ready
Current Frame ID: 179

Name: Area3
Type: Area
[✓] Show Inducer Name

Inducer Name	Type
✓ Area1	Area
✓ Area2	Area
✓ Area3	Area

Inducer: Area1
Algorithm: Change Detection
No. of models: 5
Min. no. BG models: 3
FG adaptation rate: 24
Brightness: 70.0
Region size: 5
Hue: 35.0
Threshold: 0.30
[ ] Show Change Map

Fig. 2. The user interface (UI) in our prototypic system.

In the user interface, the main window on the top left shows the current video frame, while the control panel on the right (including the *video capture*, *inducer control*, and *statistic control* subpanels) provides a mean for users to adjust related parameters. The sub-window on the bottom is the *event triggering panel* showing the detected activities at each inducer over time. In practice, here are the steps in using this user interface:

(1) Initialize a video input, either from a live video stream or a pre-captured video (using the video capture subpanel);
(2) Create statistic inducers on the video screen by drag and drop, and set related parameters in the inducer control subpanel;
(3) Lastly, we can adjust related parameters (for each statistic inducer) for the video analysis component, and also for the statistic generation callbacks using the statistic control subpanel; after that, we can start statistic generation.

2.2. *Statistic Inducers*

To flexibly support statistic generation in different scenarios, we proposed four different types of statistic inducer in the user interface, see Figure 3 for their shapes, and also the inducer control subpanel in Figure 2. Note that to create an inducer on the video screen, we just need to click on the inducer type button on the inducer control subpanel and drop an inducer of the selected type on the video screen; after that, we can interactively adjust it to fit to the related spatial location on the video screen.

Point inducer – A point inducer is an omnidirectional inducer useful for detecting non-directional changes (or activities) happening at its location; to define this inducer, users just need to click on the video screen (after the inducer type button) to specify one single point and also its size in pixel unit using the inducer control subpanel.

Line inducer – To define a line inducer, users can specify two locations on the video screen and input also the width of the line segment; but unlike the point inducer, this inducer could serve as a directional inducer aiming at detecting changes (or activities) happening perpendicular to the line segment, for example, cars crossing it on the road, see Figure 3.

Area inducer – An area inducer is a non-directional inducer like a point inducer, but could flexibly occupy an area of arbitrary size and shape on the video screen. To specify this inducer, users can click on the video screen multiple times to define the corners of an area inducer. In practice, this inducer is useful for detecting changes (or activities) over a relatively larger and arbitrary area on the video screen.

Grid inducer – A grid inducer can be understood as a two-dimensional array of area inducers; to define this inducer, users just need to specify the four grid corners on the video screen and the number of grid subdivisions inside (number of rows

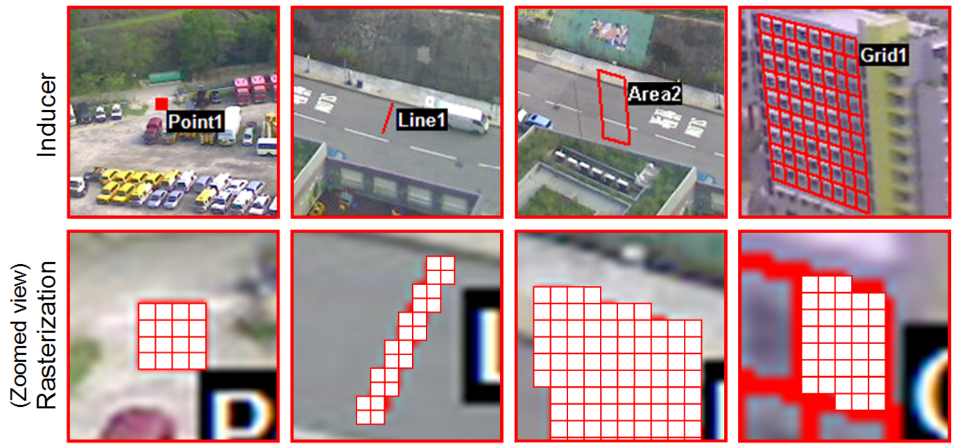


Fig. 3. Scan conversion (or Rasterization) of the statistic inducers.

and columns). In practice, this inducer is powerful not just because it allows us to quickly construct a 2D array of area inducers at one time; in addition, by using this inducer, we can efficiently generate a 2D array of time-series data, spatially corresponding to the grid cells on the video screen. Just like the example shown in the last column of Figure 3, we can use the grid inducer to detect illumination changes (e.g., light switching at night) over a matrix of windows in a dormitory. Other inducers can only generate one time-series data in nature. Furthermore, note also that we can in principle combine individual inducers to form aggregate inducers for more complicated statistic generation tasks.

After adding a statistic inducer on the video screen, the system will scan-convert the screen regions occupied by an inducer into a set of screen pixels (grouped by horizontal scanlines) so as to facilitate the upcoming video analysis process, see Figure 3 for the scan-converted examples. In details, this scan conversion (or rasterization) process employs standard scan conversion methods in computer graphics[13].

2.3. *Hierarchical Analysis Scheme*

With the statistic inducers, we not only can allow users to input prior semantic information through the user interface, we can also formulate the statistic generation process as a hierarchical scheme to facilitate a more effective video analysis. Our video analysis scheme is a three-level hierarchical analysis scheme as illustrated in Figure 4: *pixel level*, *inducer level*, and *temporal level*. To analyze an incoming image frame given in an input video stream (or a pre-captured video), the pixel-level analysis step focuses on the examination of individual pixels within each inducer, and detects potential changes individually for each of them; the inducer-level analysis step focuses on the spatial examination of these per-pixel results locally within each user-specified inducer. It then combines the detected result for each inducer; the temporal-level analysis step further compares the combined result (per inducer) against results in previous time frames and generates related raw statistic count/trigger, see the next section for details.

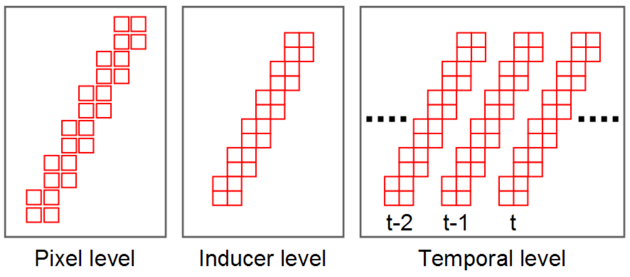


Fig. 4. The three-level hierarchical analysis scheme.

Using the hierarchical analysis scheme, we are able to cross-check information over the spatial (screen) domain as well as the temporal domain. This allows a more precise and more systematic video analysis when we formulate the statistic generation framework, and also facilitate a more efficient activity detection process by means of constraining the detection on the user-specified inducer regions.

3. Video Analysis

Before the statistic generation, a core issue we have to address in the video analysis is to detect changes happened in each user-specified inducer. This issue is basically related to the of image processing and computer vision literature, typically in the areas of *change detection* or *background modeling*. In this section, we will first present the criteria of change detection for implementing the *statistics from video* framework, and then present the unified algorithm for video analysis in terms of the hierarchical analysis scheme.

3.1. *Issues in Activity/Change Detection*

To support a practical *statistics from video* system, the video analysis component has to be able to handle the following cases, see also[28,15,8]:

(1) Foreground objects in motion
(2) Foreground objects stop and merge into the background
(3) Background objects move out of the background and become foreground objects
(4) Time of day change: gradual and global illumination change
(5) Local illumination changes (such as light switching)
(6) Shadow (by foreground objects or by moving clouds)
(7) Bootstrapping (assumes no training period)

Note that we have to address items 4 and 6 above because a practical *statistics from video* system has to handle outdoor scenes; furthermore, we have to assume continuous video capturing with adequate frame rate so that the input video can sufficiently capture object motion and also gradual illumination changes in the physical environments.

3.2. *The Video Analysis Component*

In the statistic generation framework, we base the implementation of the video analysis component on the mixture-of-Gaussian method[35,26]. Each pixel within the user-specified inducers maintains k independent Gaussian models; each has the following parameters:

- A label: Foreground (FG) or Background (BG)
- A mean color and a covariance matrix (representing the color variation in this Gaussian model)
- Confidence value (how successful this model is when representing the foreground or background)
- A foreground time counter

where k is chosen as 5 in our experiment; furthermore, to ensure the precision in pixel detection, we maintain at least 3 background models out of the 5 model slots per pixel throughout the program runtime.

Figure 5 shows the workflow of the video analysis component. The top entity labeled with "BG & FG models" denotes the sets of Gaussian models (for all pixels inside the inducers) maintained for background and foreground modeling, while the decision making procedures and the data processing procedures are labeled with numbers in circles. Altogether, there are seven sets of major procedures in the video analysis component. Procedures marked with 'b' (2*b*, 3*b*, 6*b*, and 7*b*) are for updating the Gaussian models adaptively with the video stream over time. Here, we apply the equations in[35,26] to initialize and update the Gaussian models, but since the framework is independent of the modeling approach chosen, we can essentially replace it by some other techniques.

- **Initialization** (*Procedure* 1)
  Following[26], we initialize the five Gaussian models (for each pixel inside the user-specified inducers) by using the current frame in the input image stream. Here, all initial Gaussian models are assumed to be background models with mean set to be the color at the corresponding pixel in the current frame, while the variances are set to be a user-specified value; typically, we use 10/255 for all examples shown. After that, the video analysis component can iteratively update these Gaussian model parameters using procedures 2*b*, 3*b*, 6*b*, and 7*b*, based on the new coming images from the input video stream.

- **BG and FG detection** (*Procedures* 2 *and* 3)
  When a new image is given in the next time step, each pixel inside an inducer will be testified against the existing Gaussian models to see if the incoming color

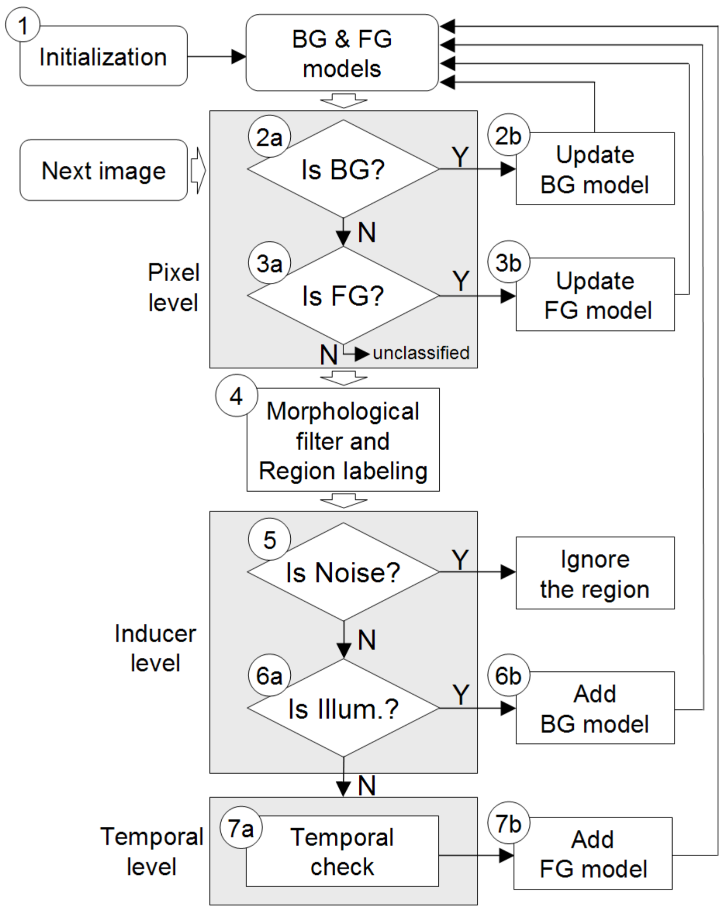


Fig. 5. Flowchart: the Video analysis component.

is representable by an existing background or foreground model. In details, the incoming pixel color is tested against each Gaussian model using the Mahalanobis distance (compare against the confidence value) in descending order of their confidence values:

$$\text{distance} \;=\; (x-\mu_{t-1})^T \; M_{t-1} \; (x-\mu_{t-1}) \;,$$

where μ_{t-1} is the mean color of a Gaussian model at previous time frame, M_{t-1} is the corresponding covariance matrix, and x is the color of the incoming pixel. If procedure $2a$ or $3a$ finds that the incoming pixel color is representable by an existing Gaussian model, the related model will be updated by procedure $2b$ or $3b$ accordingly.

- **Region growing** (*Procedure* 4)
  After we process each pixel in an inducer, we should proceed to the inducer level

for detections of unclassified pixels and foreground pixels. But before doing so, we first apply region growing to create connected regions in each inducer so that unclassified pixels and foreground pixels inside the same inducer could be connected together to form candidate foreground regions. Then, we can apply morphological filter[38] to examine the shape and size of these candidate regions.

- **Noise detection** (*Procedure* 5)
  The noise detection module is responsible for removing candidate regions with haphazard color values (noise) and also regions that are too small or thin, as reported by the morphological filter; note that users can adjust parameters (shapes and thresholds) on the filter using the statistic control panel.
- **Illumination detection** (*Procedure* 6*a*)
  After noise detection, the candidate foreground regions that are not detectable by the previous tests could be illumination changes resulted by time of day change or shadowing. Given a candidate region, the illumination detection module first converts the related pixel colors (for the unclassified pixels) to YIQ space, and check to see if the intensity changes are results of gradual illumination change or shadowing. If a certain percentage of connected pixels (by a user-defined threshold) in a candidate foreground region are found to be a result of an illumination change with similar amount of intensity changes, we will consider this candidate region as an illumination change, i.e., background, and add a new background Gaussian model for each related pixel, see "Adding a new model" below for details.
- **Temporal check** (*Procedure* 7*a*)
  After passing the illumination detection test, a candidate region will be classified as a foreground object; however, a given candidate region here could be a new foreground object just entered an inducer, or an object in motion within an inducer. In the temporal check procedure, we compare each candidate region given from procedure 6*a* against regions previously detected in the same inducer. Hence, we can estimate (by the percentage of overlapping pixels) whether a candidate region corresponds to a new object or not. In this procedure, we could also compute the related optical flow per pixel (over the candidate region) if users want to employ optical flow in the statistic generation.
- **Adding a new model** (*Procedures* 6*b and* 7*b*)
  [**Route** 6*a* **to** 6*b*] In case a candidate region is found to be a result of illumination change by procedure 6*a*, the unclassified pixels inside should be classified as background even they are not representable by any existing background Gaussian model; a new background Gaussian model has to be added for each of them. Here, one of the five existing Gaussian models has to be discarded as each pixel can only hold five Gaussian models at any given time. To select which model to be discarded, we generally pick up the one with the smallest confidence value, but to ensure a good representation of the background variation,

we maintain at least three background models per pixel (out of its five models). Hence, in case the model to be discarded is a background model and we only have three background models left, we will instead discard the model with the second smallest confidence value (and so on, in case it is also a background model). Note that the confidence value is adaptively adjusted by procedures $2b$ and $3b$, depending on whether the related Gaussian model can represent the incoming colors in previous time steps. In addition, note also that we follow[26] to update the confidence values.

[**Route** $7a$ **to** $7b$] On the other hand, in case a candidate region is found to be a foreground object (the route from $7a$ to $7b$), the unclassified pixels inside should be classified as foreground even they are not representable by any existing foreground Gaussian model; here, a new foreground Gaussian model has to be added for each of them.

In addition, to avoid misclassifying a moving object as a new foreground object, we assume sufficient frame rate in the video capture so that consecutive candidate regions of the same object can overlap with sufficient pixels. Furthermore, to allow a foreground object to merge into the background, we maintain a foreground time counter per foreground Gaussian model to keep track of the consecutive number of times a foreground model detects the same foreground object. If the counter (over the candidate region) exceeds a certain user-defined duration, we can turn the related foreground model into a background model and the foreground object (that stopped moving) will then be classified as background in the next time step.

4. Implementation and Results

4.1. *The System Architecture*

Altogether, we have the following four components in the system architecture we developed for the prototypic system, see also Figure 6:

- The *Video capture* component for capturing image frames from the video camera, and also for sending a synchronized video stream to the *User interface* component; it is implemented using the MS DirectShow API in the Windows Media SDK.

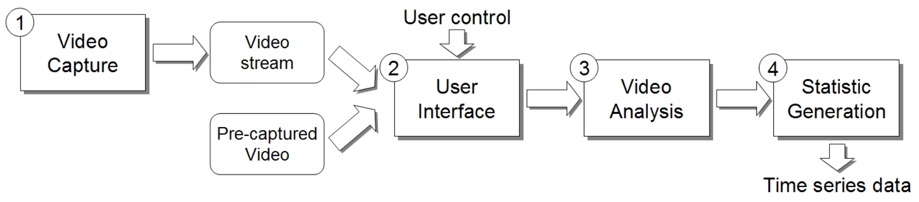


Fig. 6. The four major components in the system architecture: Video Capture, User Interface, Video Analysis, and Statistic Generation.

- The *User interface* component for the user interface display and control, so that users can efficiently set up the statistic inducers, and specify related parameters. In addition, this component also manages a set of inducer objects and passes them to the *Video analysis* component that follows. Note that other than live video streams, this component can also accept pre-captured videos as its input; it is implemented using the MS MFC API.
- As presented in Section 3, the *Video analysis* component takes a video sequence as its input, and detects changes on the inducers using the mixture-of-Gaussian model.
- Finally, the *Statistic generation* component will examine the detections generated from the video analysis component and produce statistical triggers and/or time-series data.

4.2. *The Statistic Generation component*

With the *Video analysis* component implemented as it is according to Section 3, we can attach to it a number of callback functions (per inducer), so that we can quantitatively and also procedurally customize the *Video analysis* component for statistic generation. In our current system architecture, there are two kinds of callback function provided by the *Statistic generation* component:

- The inducer-level callback allows us to thoroughly examine the detected foreground (and/or background) pixels, and quantitatively define how a raw statistic count can be triggered with respect to the corresponding inducer. In practice, we, users, can also customize this callback to analyze the quality of individual pixels in the inducer, and thoroughly examine the candidate foreground region as well. In terms of the *Video analysis* component, this callback is attached to the inducer level, and is executed (per candidate region) at the end of it, see Figure 5.
- In addition, we have the temporal-level callback to examine successive candidate regions (detected foreground objects), and evaluate whether the raw statistic count from the inducer-level is valid or not. One obvious application of it is to avoid repeated triggers on the same detection, e.g., a moving foreground object, after its first successful trigger over time. Furthermore, like the inducer-level callback, users can add in their own code to customize the raw statistic triggers for different statistic generation scenarios, e.g., an optical flow analysis or a simple intensity thresholding. In our current system architecture, the temporal-level callback is attached to the *temporal check* procedure in the *Video analysis* component, and is executed at the end of it.

Using this callback mechanism, we can gain customizable controls on the inducer and temporal levels of the *Video analysis* component, and quantitatively, and also procedurally, define how raw statistic counts can be triggered. By this means, we can quickly tailor-make the system for specific statistic generation tasks.

In addition, note also that each user-attached callback has a unique ID, and can be attached to one or more inducers on the video screen. Furthermore, users can also format the time-series result that streams out of each temporal-level callback in the *Statistic generation* component, and output it with related parameters like the time of triggering, screen pixel location, etc.

4.3. *The Hardware Setup*

Here we list the hardware equipment we employed, see Figure 7 for snapshots.

- A desktop computer: A Pentium IV PC
  (2.4GHz CPU and 512MB memory)
- A low-cost webcam: Logitech Notebook Pro
  (640×480 pixels at 15fps)

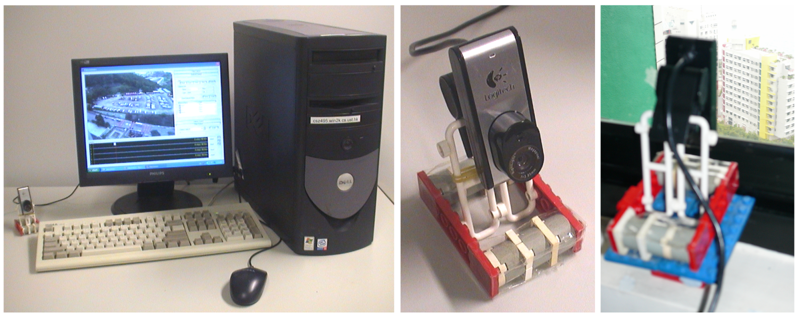


Fig. 7. The hardware setup: the whole setup (left) and the webcam (right).

Note that the connection between the PC and webcam is a standard USB 2.0 connection. With the *Video capture*, *Video analysis*, and *Statistic Generation* components all running at the same time on the above setup, we can process a video stream at 8-10 fps, which demonstrated to be sufficient in the following two testing cases: *Dormitory light switching* and *Road traffic*.

4.4. *Usability Study – Case 1: Dormitory light switching*

In the first testing case, we look at light switching events in two student dormitories, from 7pm to 6am, on Tuesday, Saturday, and Sunday nights. Here, we demonstrate the effectiveness in using grid inducers to gather statistics over matrices of windows; using the proposed graphical user interface (GUI), one can efficiently setup inducers and the related parameters. In this testing case, we only take around three minutes to set up the inducers, and the statistic generation can be started immediately afterwards.

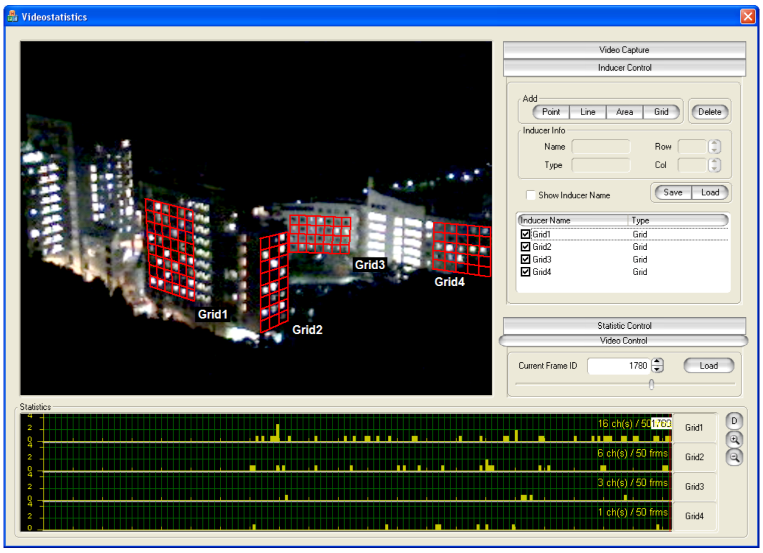


Fig. 8. The GUI setup for gathering statistics of light switching events.

Figure 8 displays the overall GUI and the inducers employed. Here, we have four grid inducers, namely GRID1, GRID2, GRID3, and GRID4, over four matrices of dormitory windows. The bottom part of the GUI is the event triggering panel (over time) showing the triggers for the active inducers, while the red line on it (right hand side) indicates the current frame being analyzed; note that the corresponding frame number is shown on top of it. Furthermore, the inducer-level triggers are depicted as horizontal yellow bars on the event triggering panel, while the numbers colored in yellow (on the right hand side) indicate the overall number of temporal-level triggers over the previous 50 time frames, for example, the label "`16 ch(s)/50 frms`" on the top row of the event triggering panel in Figure 8 shows that there were 16 detected changes over the previous 50 time frames. Furthermore, note also that when using grid inducer, we can automatically sum up all events per time frame

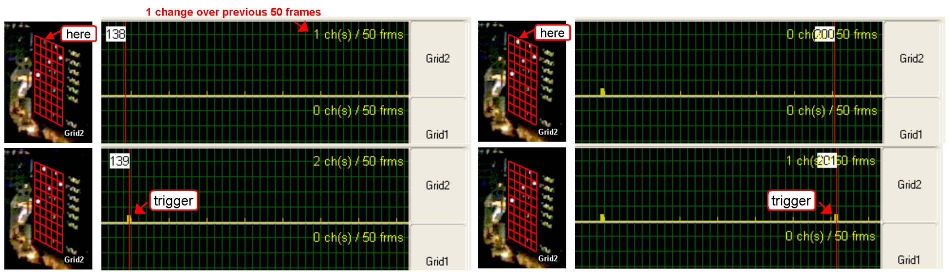


Fig. 9. An example light switching event being triggered: a light was switched on at 8:58pm (left) and then switched off at 9:29pm (right). In the event triggering panel, the horizontal axis marks the time, where one time interval refers to one minute in this testing case; on the other hand, the vertical axis shows the number of triggers per time interval.

over the entire inducer, and show them as a whole on the event triggering panel (as well as in the final statistic generation).

Figure 9 shows a particular example in this testing case. The two subfigures on the left shows a light being switched on (second window from the top left) at 8:58pm (frame ID: 139), and a corresponding trigger (the first trigger) on the event triggering panel. After a short period of time, the light was turned off at 9:29pm (frame ID: 201); the switching was detectable, and displayed on the event triggering panel (the second trigger).

Table 1. Statistics generated – Dormitory light switching.

	Tue				Sat				Sun			
Grid	**#1**	**#2**	**#3**	**#4**	**#1**	**#2**	**#3**	**#4**	**#1**	**#2**	**#3**	**#4**
7 pm - 8 pm	25	8	1	12	1	3	0	1	4	3	0	3
8 pm - 9 pm	19	11	4	2	4	4	0	0	6	4	1	4
9 pm - 10 pm	18	10	1	6	6	2	0	1	2	5	0	4
10 pm - 11 pm	11	6	1	16	3	5	1	4	3	5	1	2
11 pm - 0 am	18	7	0	2	5	2	3	3	5	5	2	3
0 am - 1 am	15	8	0	11	4	2	0	5	5	5	1	6
1 am - 2 am	17	3	3	21	4	3	3	2	5	3	1	4
2 am - 3 am	19	3	2	4	2	4	0	1	6	3	2	3
3 am - 4 am	6	1	1	3	2	1	1	3	4	2	2	1
4 am - 5 am	1	2	1	3	1	0	0	1	3	4	3	0
5 am - 6 am	2	0	0	0	4	0	2	2	1	0	1	1
Total	**151**	**59**	**14**	**80**	**36**	**26**	**10**	**23**	**44**	**39**	**14**	**31**

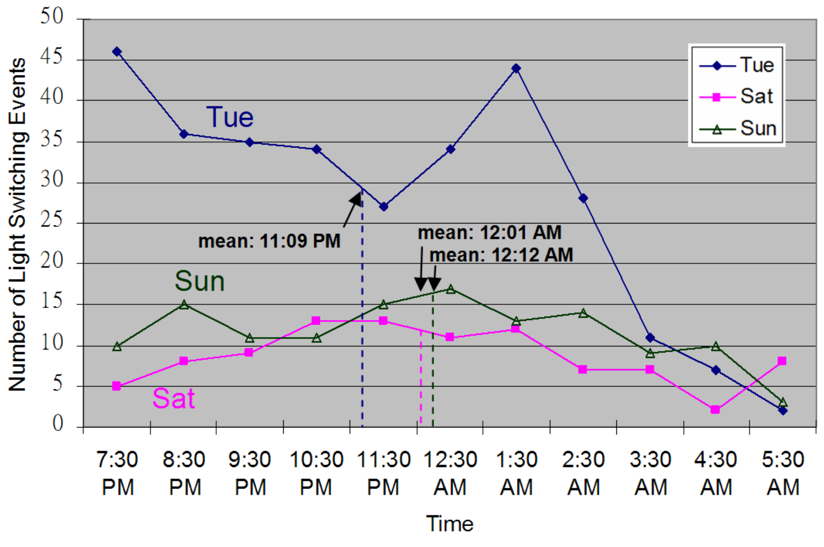


Table 1 summarizes the statistics we obtained; the table on the top presents the number of light switching events per hour over the four grid inducers on Tuesday, Saturday, and Sunday nights, while the last row in the table presents the total number of switching events. Since most students returned home on Saturday and Sunday nights, the dormitory utilization is relatively lower during the weekend, as experimentally shown in the table.

Furthermore, we also compute the mean light switching time for Tuesday, Saturday, and Sunday nights, and depict the results in the graph below Table 1. Here we found that the mean light switching time for Tuesday, Saturday, and Sunday nights are 11:09pm, 12:01pm, and 12:12pm, respectively; thus, we believe that on average, students tend to sleep later on Saturday and Sunday nights. In addition, it is worth to note that this experiment can be further customized to generate many different kinds of interesting statistics, for example, we can modify the callback function to trigger only lights being switched off (or lights being switched on); using the proposed GUI design, we are able to efficiently generate timely statistics with great ease.

4.5. *Usability Study – Case 2: Road traffic statistics*

The second testing case looks at road traffic, and generates statistics about the cars passing through an array of area inducers. The testing period is from 11:00am to 14:30pm, and four inducers, namely AREA1, AREA2, AREA3, and AREA4, are created on the video screen, see Figure 10 for a short image sequence extracted from the video stream; it shows a particular car moving from left to right. The car appeared in the first sub-figure at 12:56pm (frame ID: 27294); when it moved into the region of AREA1 (frame ID: 27299), a change was triggered and shown correspondingly as a yellow bar on the event triggering panel of AREA1. When the car reached the second inducer (AREA2) (frame ID: 27305), another change was triggered at AREA2, see the third sub-figure. The statistic triggering continued until the car moved to the rightmost side, see the last three sub-figures.

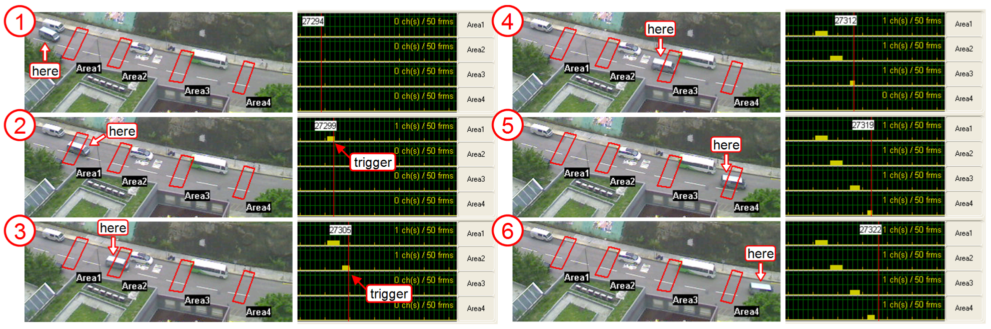


Fig. 10. A statistics from video example: studying road traffic using a series of four area inducers, see also Figure 2 for the entire view of the physical environment being captured.

Furthermore, over a long period of testing time, some cars may just stop at an inducer and park inside. In this case, the car has to be merged into the background to avoid excessive triggers. Figure 11 demonstrates a moving car parked inside inducer AREA4. When the car entered inducer AREA4, a change was triggered, see the middle sub-figure. After that, the car stayed in it and the (inducer-level) trigger continued. Shortly after the time period (recorded by the foreground time

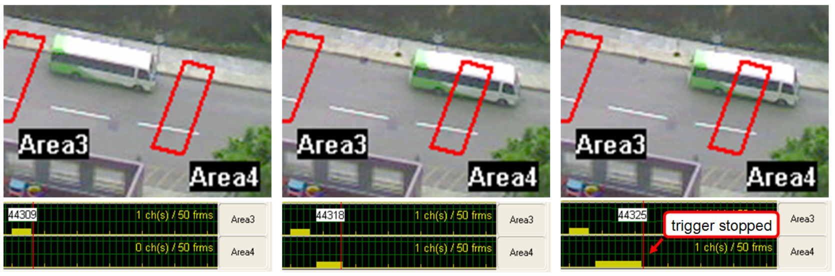


Fig. 11. A car stopped and merged into the background.

counter) exceeded a certain user-specified duration, the trigger stopped because of the foreground adaptation mechanism. Note that in the temporal-level callback, we produce only one (temporal-level) statistic trigger for a series of consecutive inducer-level triggers because all these inducer-level triggers (the yellow bar shown on the event triggering panel) are resulted by the same foreground object.

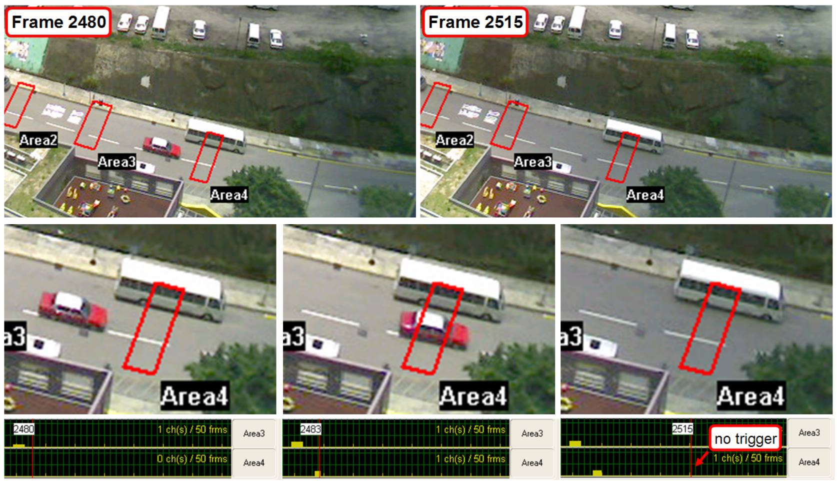
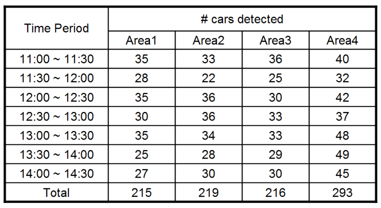


Fig. 12. Illumination detection: a moving cloud in the scene.

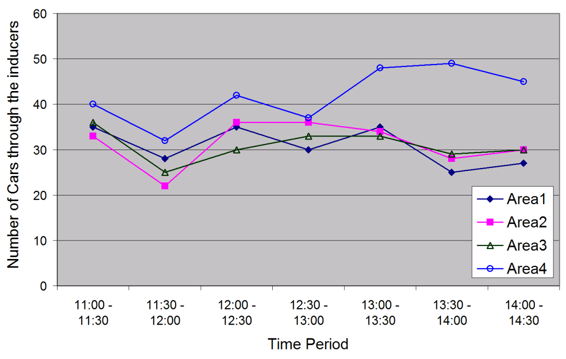
In outdoor environments, the illumination condition could change gradually with time, or shadowed by some moving clouds; the video analysis component has to adapt to this so as to avoid false detection. Figure 12 shows a red car crossing inducer Area4. Changes were triggered only when the car overlapped with Area4; after the car passed through the inducer, no more trigger was produced at Area4 even though the road was shadowed by a cloud moving from right to left.

Table 2 presents the related statistic results. We can see from the table that the number of cars detected at all inducers are nearly the same except for inducer Area4. There are two reasons we observed. Firstly, noting that the road in this

Table 2. Statistics – Road Traffic.



Time Period	# cars detected			
	Area1	Area2	Area3	Area4
11:00 ~ 11:30	35	33	36	40
11:30 ~ 12:00	28	22	25	32
12:00 ~ 12:30	35	36	30	42
12:30 ~ 13:00	30	36	33	37
13:00 ~ 13:30	35	34	33	48
13:30 ~ 14:00	25	28	29	49
14:00 ~ 14:30	27	30	30	45
Total	215	219	216	293



testing case is just one-way to the car park shown in Figure 2, some drivers just made U-turns around AREA4 without going through the other inducers. Secondly, some cars could be counted twice because of the foreground adaptation; a car is counted for the first time when it enters an inducer, and could be counted again when it moves away after a long period of parking. In our current system, we can memorize connected foreground objects that adapted to be background, so that when a new detection is found, we can compare the related pixel regions for the new detection against the memorized foreground objects and avoid double counting. In addition, user can also specify a time duration threshold to define how long we should memorize these adapted foreground objects in our system. Experimentally, the success rates we found for using line inducers and area inducers in this case study are around 93% and 96%, respectively. The detection failure is mostly due to: 1) some cars moved across the inducers at a very fast speed and they did not even overlap with the inducers (in particular, line inducers), 2) some pedestrians moved across the road at the inducer location and generated triggers, and 3) the color of some cars (grey or light black) were too similar to that of the road and these cars were missed in the detection.

[As the paper can only present static images, reviewers are invited to look at the companion video for a more natural inspection on the user interface control and the statistic generation results.]

5. Conclusion and Discussion

This paper introduced a user interface approach for efficiently generating timely statistics from video. Through the carefully-designed graphical user interface, users can readily input prior semantic information by using the statistic inducers to mark up relevant activities in the corresponding physical environment. Then, we can precisely direct and constrain the video analysis process on the user-marked areas, and efficiently customize the system to generate related statistics accordingly. Here in the proposed system, it only takes users a few minutes of time to set up the statistic inducers; and once it is done, timely statistic data can be immediately generated by the system in a fully automatic manner. To demonstrate the effectiveness of this idea, we implemented the prototypic system shown in Section 4, and experimented two testing cases: *Dormitory light switching* and *Road traffic*. These two cases demonstrated that timely statistics can be readily generated from the input videos in a highly user friendly manner.

Future work The mechanism of generating statistics from videos opens a wide range of research and development opportunities. First, we can investigate the use of multiple synchronized video streams for generating inter-related statistics, spatially referenced to different locations in the physical environment. On the other hand, we can explore the use of generated statistics to design high-level spatial data queries from the perspective of database research. Furthermore, we can also investigate various kinds of system customizations for different potential applications, such as, sports, shopping pattern, and marketing.

Acknowledgments

We would like to thank C.-K. Tang for his advice in video analysis, Shirley Ng for implementing the video capture component, and Felix Lo for composing the companion video.

References

1. Til Aach, Lutz Dümbgen, Rudolf Mester, and Daniel Toth. Bayesian illumination-invariant motion detection. In *Proceedings of IEEE International Conference on Image Processing*, volume 3, pages 640–643, 2001.
2. Til Aach and Audré Kaup. Bayesian algorithms for adaptive change detection in image sequences using Markov random fields. *Signal Processing: Image Communication*, 7(2):147–160, Aug. 1995.
3. Til Aach, Audré Kaup, and Rudolf Mester. Statistical model-based change detection in moving video. *Signal Processing*, 31(2):165–180, Mar. 1993.
4. Aseem Agarwala, Aaron Hertzmann, David H. Salesin, and Steven M. Seitz. Keyframe-based tracking for rotoscoping and animation. *ACM Transactions on Graphics*, 23(3):584–591, 2004.
5. Kiran S. Bhat, Steven M. Seitz, Jessica K. Hodgins, and Pradeep K. Khosla. Flow-based video synthesis and editing. *ACM Transactions on Graphics*, 23(3):360–363, 2004.
6. Andrea Cavallaro and Touradj Ebrahimi. Change detection based on color edges. In *Proceedings of the IEEE International Symposium on Circuits and Systems (ISCAS-2001)*, pages 141–144, 2001. vol. 2.
7. Andrea Cavallaro, Olivier Steiger, and Touradj Ebrahimi. Multiple video object tracking in complex scenes. In *Proceedings of the tenth ACM international conference on Multimedia*, pages 523–532, 2002.
8. Robert Collins, Alan Lipton, Takeo Kanade, Hironobu Fujiyoshi, David Duggins, Yanghai Tsin, David Tolliver, Nobuyoshi Enomoto, and Osamu Hasegawa. A system

for video surveillance and monitoring. Technical Report CMU-RI-TR-00-12, Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, May 2000.

9. Dorin Comaniciu, Visvanathan Ramesh, and Peter Meer. Real-time tracking of non-rigid objects using mean shift. In *Proceedings of International Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 142–149, 2000. vol. 2.
10. Gareth Daniel and Min Chen. Video visualization. In *Proceedings of IEEE Visualization 2003*, pages 409–416, 2003.
11. Emrullah Durucan and Touradj Ebrahimi. Change detection and background extraction by linear algebra. In *Proceedings of the IEEE*, volume 89, pages 1368–1381. IEEE, 2001.
12. Ahmed Elgammal, Ramani Duraiswami, David Harwood, and Larry S. Davis. Background and foreground modeling using nonparametric kernel density estimation for visual surveillance. In *Proceedings of the IEEE*, volume 90, pages 1151–1163, 2002.
13. James D. Foley, Andries van Dam, Steven K. Feiner, and John F. Hughes. *Computer graphics: principles and practice (2nd ed.)*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1990.
14. Greg D. Hager and Peter N. Belhumeur. Real-time tracking of image regions with changes in geometry and illumination. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 403–410, June 1996.
15. Ismail Haritaoglu, David Harwood, and Larry S. Davis. W4: real-time surveillance of people and their activities. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8):809–830, 2000.
16. Li-Wei He and Zhengyou Zhang. Real-time whiteboard capture and processing using a video camera for remote collaboration. *IEEE Transactions on Multimedia*, 2006. accepted for publication.
17. Klaus-Peter Karmann and Achim von Brandt. Moving object recognition using an adaptive background memory. *Time-Varying Image Processing and Moving Object Recognition*, 2:289–296, 1990. Elsevier, Amsterdam, The Netherlands.
18. Jean-Francois Lalonde, Derek Hoiem, Alexei A. Efros, Carsten Rother, John Winn, and Antonio Criminisi. Photo clip art. *ACM Transactions on Graphics*, 26(3):Article no. 3, 2007.
19. Yvan G. Leclerc, Quang-Tuan Luong, and Pascal Fua. Self-consistency and MDL: A paradigm for evaluating point-correspondence algorithms, and its application to detecting changes in surface elevation. *International Journal of Computer Vision*, 51(1):63–83, 2003.
20. Yin Li, Jian Sun, and Heung-Yeung Shum. Video object cut and paste. *ACM Transactions on Graphics*, 24(3):595–600, 2005.
21. Robert L. Lillestrand. Techniques for change detection. *IEEE Transactions on Computers*, 21(7):654–659, Jul. 1972.
22. William A. Malila. Change vector analysis: An approach for detecting forest changes with Landsat. In *Proceedings of the 6th Annual Symposium on Machine Processing of Remotely Sensed Data*, pages 326–335, 1980.
23. S. Negahdaripour. Revised definition of optical flow: Integration of radiometric and geometric cues for dynamic scene analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(9):961–979, 1998.
24. Jiazhi Ou, Xilin Chen, Susan R. Fussell, and Jie Yang. DOVE: Drawing over video environment. In *Proceedings of the eleventh ACM international conference on Multimedia*, pages 100–101, 2003.
25. Gopal S. Pingali, Agata Opalach, Yves Jean, and Ingrid Carlbom. Visualization of sports using motion trajectories: providing insights into performance, style, and strat-

egy. In *Proceedings of IEEE Visualization 2001*, pages 75–82, 2001.
26. Fatih Porikli and Oncel Tuzel. Bayesian background modeling for foreground detection. In *Proceedings of the third ACM international workshop on Video surveillance and sensor networks*, pages 55–58, 2005.
27. Yingge Qu, Tien-Tsin Wong, and Pheng-Ann Heng. Manga colorization. *ACM Transactions on Graphics*, 25(3):1214–1220, 2006.
28. Richard J. Radke, Srinivas Andra, Omar Al-Kofahi, and Badrinath Roysam. Image change detection algorithms: a systematic survey. *IEEE Transactions on Image Processing*, 14(3):294–307, March 2005.
29. Jens Rittscher, Jien Kato, Sebastien Joga, and Andrew Blake. A probabilistic background model for tracking. In *Proceedings of the 6th European Conference on Computer Vision (ECCV)*, volume 2, pages 336–350, 2000.
30. Augusto Román, Gaurav Garg, and Marc Levoy. Interactive design of multi-perspective images for visualizing urban landscapes. In *Proceedings of IEEE Visualization 2004*, pages 537–544, 2004.
31. Paul L. Rosin. Thresholding for change detection. In *Proceedings of British Machine Vision Conference*, pages 212–221, 1997.
32. Paul L. Rosin and Efstathios Ioannidis. Evaluation of global image thresholding for change detection. In *Pattern Recognition Letter*, volume 24, pages 2345–2356, 2003.
33. Peter Sand and Seth Teller. Video matching. *ACM Transactions on Graphics*, 23(3):592–599, 2004.
34. Ashbindu Singh. Digital change detection techniques using remotely-sensed data. *International Journal of Remote Sensing*, 10(6):989–1003, 1989.
35. Chris Stauffer and W. E. L Grimson. Adaptive background mixture models for real-time tracking. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, volume II, pages 246–252, 1999.
36. Luigi Di Stefano, Stefano Mattoccia, and Martino Mola. A change-detection algorithm based on structure and colour. In *Proceedings of IEEE Conference on Advanced Video and Signal Based Surveillance*, pages 252–259, 2003.
37. Bjoern Stenger, Visvanathan Ramesh, Nikos Paragios, Frans Coetzee, and Joachim M. Bouhman. Topology free Hidden Markov models: Application to background modeling. In *Proceedings of the International Conference on Computer Vision*, pages 294–301, 2001.
38. Elena Stringa. Morphological change detection algorithms for surveillance applications. In *Proceedings of the 11th British Machine Vision Conference*, pages 402–411, 2000.
39. Daniel Toth, Til Aach, and Volker Metzler. Illumination-invariant change detection. In *Proceedings of the 4th IEEE Southwest Symposium on Image Analysis and Interpretation*, pages 3–7, 2000.
40. Kentaro Toyama, John Krumm, Barry Brumitt, and Brian Meyers. Wallflower: principles and practice of background maintenance. In *Proceedings of IEEE International Conference on Computer Vision*, volume 1, pages 255–261, 1999.
41. Meredith S. Ulstad. An algorithm for estimating small scale differences between two digital images. *IEEE Transactions on Computers*, 5:323–333, 1973.
42. Anton van den Hengel, Anthony Dick, Thorsten Thormählen, Ben Ward, and Philip H. S. Torr. VideoTrace: rapid interactive scene modelling from video. *ACM Transactions on Graphics*, 26(3):Article no. 86, 2007.
43. Jue Wang, Pravin Bhat, R. Alex Colburn, Maneesh Agrawala, and Michael F. Cohen. Interactive video cutout. *ACM Transactions on Graphics*, 24(3):585–594, 2005.
44. Jue Wang, Bo Thiesson, Yingqing Xu, and Michael Cohen. Image and video segmen-

tation by anisotropic kernel mean shift. In *Proceedings of European Conference on Computer Vision (ECCV)*, pages 238–249, 2004.
45. Jue Wang, Yingqing Xu, Heung-Yeung Shum, and Michael F. Cohen. Video tooning. *ACM Transactions on Graphics*, 23(3):574–583, 2004.
46. Christopher Richard Wren, Ali Azarbayejani, Trevor Darrell, and Alex Paul Pentland. Pfinder: Real-time tracking of the human body. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(7):780–785, 1997.
47. Tai-Pang Wu, Chi-Keung Tang, Michael S. Brown, and Heung-Yeung Shum. ShapePalettes: interactive normal transfer via sketching. *ACM Transactions on Graphics*, 26(3):Article no. 44, 2007.
48. Wen Wu, Xilin Chen, and Jie Yang. Incremental detection of text on road signs from video with application to a driving assistant system. In *Proceedings of the 12th ACM international conference on Multimedia*, pages 852–859, 2004.
49. Changsheng Xu, Xi Shao, Namunu C. Maddage, and Mohan S. Kankanhalli. Automatic music video summarization based on audio-visual-text analysis and alignment. In *Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 361–368, 2005.

Photo and Bibliography

Man-Kang Leung received the BSc and MPhil degrees in computer science from the Hong Kong University of Science and Technology in 2004 and 2006, respectively.

His research interests include texture synthesis, tile-based modeling and rendering methods, the bidirectional texture functions (BTF), geometric modeling, and multimedia applications concerning video processing.

Chi-Wing Fu received the BSc and MPhil degrees in computer science from the Chinese University of Hong Kong in 1997 and 1999, respectively, and the PhD degree in computer science from Indiana University at Bloomington in 2003.

He is now a visiting assistant professor in the Department of Computer Science at the Hong Kong University of Science and Technology. His research interests include image-based modeling and rendering, texture synthesis, medical visualization, and visualization and navigation in large-scale astrophysical environments. He is a member of the IEEE, the IEEE Computer Society, and ACM SIGGRAPH.