Virtual Postman – Real-Time, Interactive *Virtual Video*

Alan J. Lipton
The Robotics Institute, Carnegie Mellon University, 5000 Forbes Ave, Pittsburgh, PA, 15213
email: ajl@cs.cmu.edu, URL: http://www.cs.cmu.edu/~ vsam

Abstract

This paper presents a new paradigm for data interaction called virtual video. *The concept is that video streams can be interactively augmented in real-time by both addition and removal of visual information. Removed information must be seamlessly replaced with relevant "background" information. Added information can be in the form of computer graphics, as in conventional augmented reality, or imagery derived from the video stream itself. Adding video-derived imagery means that a "real" character can be simulated in different places or times in the video stream, or interacting (fictitiously) with other characters. Achieving this requires an understanding of the motion and appearance of the target character so that it can be realistically inserted.*

Video understanding technology is now sufficiently advanced to make interactive, real-time virtual video *a possibility. An example is given in the form a computer game called* Virtual Postman *in which moving characters detected in a video stream can be "killed" and thus removed from that stream. Furthermore, characters detected in the video stream are stored, analysed for rigid or periodic motion and then smoothly re-inserted into the video stream at arbitrary places, times and scales as "clones" of live characters, or "zombies" of dead ones.*

Keywords: *Computer vision, image analysis, virtual video, augmented reality.*

1 Introduction

Augmented reality has been a research topic in the vision community for some time. The notion is that video imagery can be augmented by accurately registered computer graphics. Computerised X-Ray vision[3], or video assisted surgery are two examples of this. However, as the field of video understanding[13][12] matures, it becomes increasingly possible to analyse and interact with real-time video-derived data directly.

Virtual video (as coined by the author) is the idea that video streams can be interactively altered in real-time so that they can be treated as virtual worlds into which objects can be interactively inserted or removed at will. Furthermore, augmentations to the video stream can be derived directly from the video stream, rather than being solely computer generated. Thus, "real" objects can appear to move through space or time in a synthetic manner. An example would be moving a building from one place to another in a video scene; or have a person appear 10 minutes after s/he actually walked through the scene. A more complex example is to create a synthetic character based on the actions and appearance of a real one and, at some point, seamlessly replace the real with the synthetic.

Applications for this technology are legion in fields such as video-teleconferencing, surveillance, and entertainment. This paper presents an illustrative example of a *virtual video* system in the form of a computer game called *Virtual Postman*. The significant point is that a real-time video stream can itself become a playing field on which the player interacts directly with both real and synthetic objects.

In *Virtual Postman*, a camera is pointed at a generic scene, either indoor or outdoor, and the video stream is viewed by a player on a desktop computer. Moving objects such as vehicles and people are detected and presented to the player as "targets". The player can then simulate shooting the targets which appear to expire in computer generated explosions. "Dead" targets are synthetically removed from the video stream in real-time. Furthermore, these targets can, at random, synthetically be brought back to "life" as zombies enhanced by computer graphics and re-inserted into the video stream at any position or time. This concept is similar to one suggested by Scott Adams (of Dilbert(tm) fame) in his latest book[1].

1.1 Video in Games

One of the long-time goals of the entertainment industry is the creation of realism. To achieve this, the movie industry has made a great investment in computer graphics to create realistic false images. Conversely, the computer game industry has been integrating photo-realistic still imagery and video to enhance the player's experience. To date, this integration has been largely non-interactive using only "canned" video sequences to achieve little more than setting atmosphere.

Early examples of the use of imagery in games used still images or canned video sequences as a backdrop to the action, with computer generated characters overlaid, not really partaking of the imagery. A slightly more interactive use of video is displayed in more recent games such as Return to Zork(tm) and Myst(tm) in which short, relevant video sequences provide the player with timely information or atmosphere. The most interactive use of video has been in video-disc based games such as Dragon's Lair(tm), in which the game itself is made up of small image sequences, each containing a small problem or challenge. Based on the player's choice, the next appropriate video sequence is selected (exploiting the fast random access time available to the video-disc medium) providing the next challenge.

There has been some effort made to use video interactively, most notably as an input device. There exist companies that produce games based on blue screen technology. Real players are inserted into a virtual environment to perform simple actions like tending a virtual soccer goal, or shooting virtual baskets. These games require considerable infrastructure. The player must wear distinguishing clothing such as green gloves so that the computer can recognise body parts, and the game is played out on a large blue screen stage. More modest applications of this type run on desktop computers such as SGI's Lumbus(tm) in which the IndyCam is used for simple head or hand tracking to control a plant-like creature called a "Lumbus" in 3D. Using *virtual video* provides a means, for the first time, of using real-time, live video interactively as a game playing field.

2 A *Virtual Video* Architecture

The two fundamental challenges of *virtual video* are the ability to **seamlessly remove characters from a video stream** and **the ability to seamlessly add synthetic characters to a video stream**. Note that synthetic characters can be derived from the video stream itself and, thus, their motion must be understood in order to re-create them accurately in different times and places. Figure 1 shows a potential architecture for a *virtual video* application. An input video stream is segmented into background and foreground regions which can be archived for later use. A *virtual video* image can be built by combining background, foreground, and synthetic components into a coherent image. Synthetic components may be derived from the video stream at some previous time (such as background data which must be used to "fill in the blanks" left behind by some "dead" character, or previously seen characters which are being brought back from the dead); or they may be completely computer generated.

Some of the specific research challenges of a *virtual video* architecture are: segmentation of a video stream into a background and a set of foreground characters; tracking foreground characters and disambiguating them as they interact with each other and the background; removing characters from the video stream and seamlessly replacing them with appropriate background (or other foreground) imagery; and analysing the motion of real characters so they may be inserted as required as realistic synthetic characters. Furthermore, all of this must be done automatically in real-time by appropriate computer vision techniques!

For this application, foreground objects are considered to be moving characters such as people and vehicles. There is a great store of techniques for extracting moving objects from imagery. Some approaches are model-based, such as [14][20], that look for specific types of objects. More general motion detection schemes are outlined in [4][9]. The most promising approaches use dynamically adaptive background subtraction[5][7]. These are preferred for two reasons: they provide the most complete extraction of moving objects; and a by-product of motion detection is a model of the scene background which can be used to replace "dead" foreground characters! The method used here is taken from[5] and is described briefly in section 3.

Tracking characters is an important component of the *virtual video* architecture. The system must be able to distinguish all the different characters in a scene to ensure that a "dead" character does not accidentally become reinstated. To achieve this, it is necessary to track characters through occlusion with other characters, or background objects. A great deal of work has been done on tracking. The most common current methods in computer vision are Kalman filters[11] and the CONDENSATION algorithm[10]. In fact one of the claims to fame of CONDENSATION is its ability to track multiple objects through occlusion. However, in this application, given that video-derived characters are being tracked, a template matching solution[15] is used and described in section 4. An extension to the tracking algorithm for tracking multiple objects through occlusion is described in [16]. One by-product of this tracking scheme is that, as characters are tracked, templates can be collected showing visual variability over time. These image sequences are useful for synthetic character generation.

There are several situations when it is necessary to insert synthetic characters into the *virtual video* stream in the context of *Virtual Postman*. Obviously, when a "dead" character is brought back to life, it must appear to interact with the environment in a realistic manner. A more subtle situation is when a "live" character is occluded by a "dead" one. Here, there is no imagery in the video stream to represent the character, so synthetic imagery must be inserted to connect the real segments without apparent discontinuity. To achieve this, it is necessary to model the appearance of the character's motion. Modeling the motion of humans and vehicles is a topic of great interest to both the vision and graphics community. Gavrila[6] provides an excellent sur-

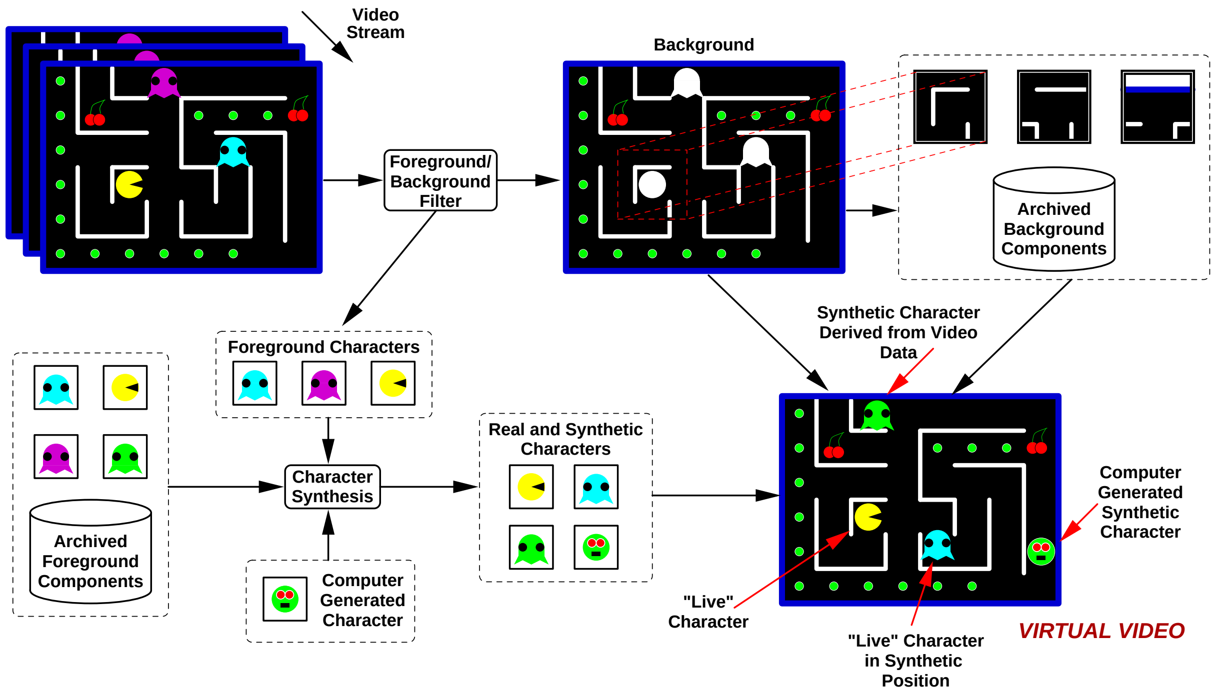


Figure 1. *Overview of the* Virtual Postman virtual video *architecture. Video images are segmented into background and foreground components. Background data is archived so it can be used to replace "dead" foreground objects. Foreground objects are archived so they can be re-inserted as synthetic objects modified by computer graphics. The* virtual video *image is integrated by combining background components, "live" characters, and synthetic characters.*

vey of human motion analysis techniques. For this simple application, it is assumed that vehicles are rigid and move in straight lines, and humans (or animals) are non-rigid and move with periodic motion. Thus it is only necessary to determine the rigidity and periodicity of a character. View invariant approaches to this problem have been attempted by image matching[18][17][19]. Fujiyoshi and Lipton[5] use image skeletons to determine walking and running of humans. In this application, rigidity is determined by examining internal optic flow[16] of a character and periodicity is determined by an image matching method similar to [18] as described in section 5. This paper proposes, in section 6, a periodicity model for a non-rigid object which consists of an image sequence representing a complete cycle of the object's motion and a displacement sequence representing the spatial relationship between each image. These *periodic sequences* can then be repeated, scaled, physically moved, or inverted to simulate the object appearing in any position at any time.

3 Detection of Moving Objects

Detection of *blobs* in a video stream is performed by a process of background subtraction using a dynamically updated background model[5] from a stable video stream (either from a static camera, or stabilised by a suitable

Figure 2. *Moving* blobs *are extracted from a video image.*

algorithm[8]) denoted $I_n(x)$ where I is the intensity of pixel $x = (i, j)$ at frame n.

Firstly, each frame is smoothed with a 3×3 Gaussian filter to remove video noise. The background model $B_n(x)$ is initialised by setting $B_0 = I_0$. After this, for each frame, a binary motion mask image $M_n(x)$ is generated containing all moving pixels

$$M_n(x) = \begin{cases} 1, & |I_n(x) - B_{n-1}(x)| > T \\ 0, & |I_n(x) - B_{n-1}(x)| \leq T \end{cases} \quad (1)$$

Where T is an appropriate threshold. After this, non-moving pixels are updated using an IIR filter to reflect changes in the scene (such as illumination – making the method appropriate to both indoor and outdoor settings)

$$B_n(x) = \begin{cases} B_{n-1}(x), & M_n(x) = 1 \\ \alpha I_n(x) + (1-\alpha)B_{n-1}(x), & M_n(x) = 0 \end{cases} \quad (2)$$

where α is the filter's time constant parameter. Moving pixels are aggregated using a connected component approach so that individual *blobs* can be extracted. An example of these extracted *blobs* is shown in figure 2. Two things about this method of motion detection are relevant to the *Virtual Postman* application. The dynamic background model contains the most recent background image information *for every pixel* – even the ones currently occluded! Also, the moving objects extracted are complete. They contain no background pixels and no holes, so they are ideal templates to be removed and reinserted into the *virtual video* stream. Consequently, removing characters from the video stream is easily achieved by simply replacing their pixels with the corresponding background pixels from B_n.

4 Tracking Targets

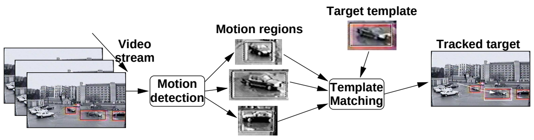


Figure 3. *The tracking process. Moving objects are detected in each video frame. Individual objects are tracked by visual template correlation matching.*

Tracking of objects is performed using the technique[15] as shown in figure 3. Extracted *blobs* are tracked from frame to frame by visual template correlation matching. However, the template matching is only applied in regions where motion is detected, to reduce the computational load and increase the method's overall reliability.

Consider a *blob* Ω_n detected in frame I_n which needs to be tracked to frame I_{n+1}. In frame, I_{n+1}, there are several *blobs* ω^k_{n+1} which could be a new instance of Ω_n. Ω_n is convolved with each of ω^k_{n+1} in turn to determine both the best match, and the new location, or displacement, of Ω.

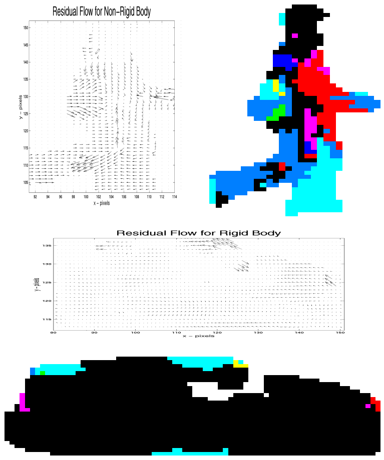
First, a rectangular windowing function $W(x)$ of size (W_i, W_j) is defined which is just large enough to enclose Ω_n. Each element of $W(x)$ is based on whether the individual pixel x is moving. That is $W(x) = M_n(x)$ within (W_i, W_j). The convolution D^k takes the form

$$D^k(x; d) = \sum_{i=1}^{i=W_i} \sum_{j=1}^{j=W_j} \frac{|W(i,j) I_n(i,j) - I_{n+1}((i,j)+d)|}{||W(x)||} \tag{3}$$

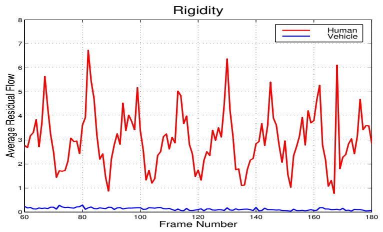
where $||W(x)||$ is a normalisation constant given by

$$||W(x)|| = \sum_{i=1}^{i=W_i} \sum_{j=1}^{j=W_j} W(i,j) \tag{4}$$

and d is a displacement. The values of d are chosen to overlap Ω_n with the set of pixels in ω^k_{n+1}.



(a)



(b)

Figure 4. *(a) The* residual flow *computed for two characters. Also, a flow-based clustering is shown. Clearly, arms and legs are apparent in the human clustering whereas only one significant cluster is visible for the vehicle. (b) The rigidity measure* $\overline{v}_R$ *calculated over time. Clearly the human displays a higher average* residual flow *and is thus less rigid. The rigidity measure also exhibits the periodic nature that would be expected for a human moving with a constant gait.*

The values of k and d are found to globally minimise $\{D^k(x; d)\}$

$$\{k_{\min}; d_{\min}\} = \min_{\{k;d\}} \{D^k(x; d)\} \tag{5}$$

From these values, the appearance of Ω_{n+1} in I_{n+1} is taken as $\omega^{k_{\min}}_{n+1}$ and its pixel displacement (or velocity if frame rate δt is known) is taken as

$$d_{\min} = \vec{v}(x)\delta t \tag{6}$$

5 Character Rigidity

Rigidity of a character is determined by its optical *residual flow* as defined in [16]. A local optic flow computation is applied to a tracked character using a method similar to Anandan's[2] to produce a flow field $\vec{v}(x)$ for the pixels in that character. The *residual flow* $\overline{v}_R$ is a measure of the

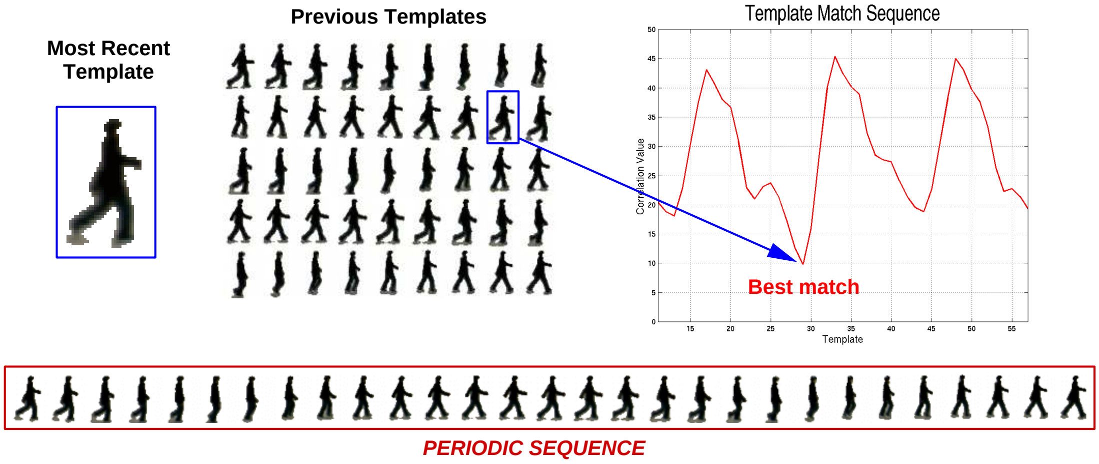


Figure 5. *Determining periodicity. The most recent template* Ω_k *is matched with previous templates and the correlation values* $C_k(y)$ *are stored. The template which minimises* $C_k(y)$ *marks the beginning of a* periodic sequence

amount of internal visual motion within the character. In this case, $\overline{v}_R$ is taken as the standard deviation of the flow vectors computed:

$$\overline{v}_R = \frac{\sum_p |\vec{v}(x) - \overline{v}|}{p} \qquad (7)$$

where p is the number of pixels in the character's image ($= ||W(x)||$ from section 4). Figure 4(a) shows the residual flow field $\vec{v}(x)$ and a simple clustering based on $\vec{v}(x)$ for two characters. The results of the *residual flow* computation are shown in figure 4(b). Clearly, this measure can be used to distinguish rigid from non-rigid objects.

6 Determining a *Periodic Sequence*

Determining periodicity is important for generating synthetic characters based on video imagery. If periodic motion is assumed, it becomes possible to extract a *periodic sequence* $P(k) = \{\Omega_k, d_k\}$ which represents the character exhibiting one cycle of motion over a set of frames $k \in [P_0, P_N]$. The sequence consists of both the visual appearance Ω of the character at each frame and the frame-to-frame velocity d as well. This sequence can then be repeated, scaled, inverted, or physically moved in the image, creating a realistic synthetic character. For a rigid character (such as a vehicle), the *periodic sequence* may contain only one element – a good view of the object Ω_0, and its pixel velocity d_0.

Periodicity is determined by a method similar to that discussed in [18]. For each instance of a character detected in the video stream a visual template is collected. The result is a list of templates $\Omega_1 \cdots \Omega_n$. To determine periodicity, each template Ω_k is matched (by equation 3) to all of the previous templates $\Omega_1 \cdots \Omega_y \cdots \Omega_{k-1}$. For each match, the minimum of the convolution $D_k^y(x;d)$ is collected as a sequence $C_k(y) = \min D_k^y(x;d)$.

The template Ω_Y which is closest in appearance to Ω_k will have the lowest value of $C_k(Y)$ and is selected as the "starting" point for the *periodic sequence* $P_0 = Y$ and $P_N = k-1$ is taken as the "end" point. Each element of the *periodic sequence* $P(y)$ contains both the template Ω_y and the corresponding frame to frame pixel velocity $d^y_{\min}$ as computed by equation 6 of section 4. This captures both the visual and temporal components of one cycle of the character's motion. Figure 5 shows the determination of a *periodic sequence* for a non-rigid character.

7 Synthesising Characters

There are situations in which it is necessary to insert synthetic characters into the *virtual video* stream. These may need to represent: real "live" characters if, for example, a "live" character is being occluded in the video stream by a "dead" one; video-derived characters at fictitious locations or times if, for example, a "dead" character is brought back to life as a zombie; completely synthetic computer generated characters; or even a combination – a computer enhanced video-derived character.

Generating zombie characters from stored *periodic sequences* $P(k)(= \{\Omega_k, d_k\})$ is fairly straightforward. These zombies can be made to move at arbitrary speeds and appear at arbitrary places or scales within the *virtual video* image. They can even be made to move in the opposite direction from the source character. They are inserted by selecting a starting frame I_{n_0} and position in the *virtual video* image x_0. The *periodic sequence* can be scaled in size by a factor K and in time by a factor T. If necessary, a flip operator $F(\Omega)$ can be applied to the images to make them move in the reverse direction. Then, at each subsequent frame I_n,

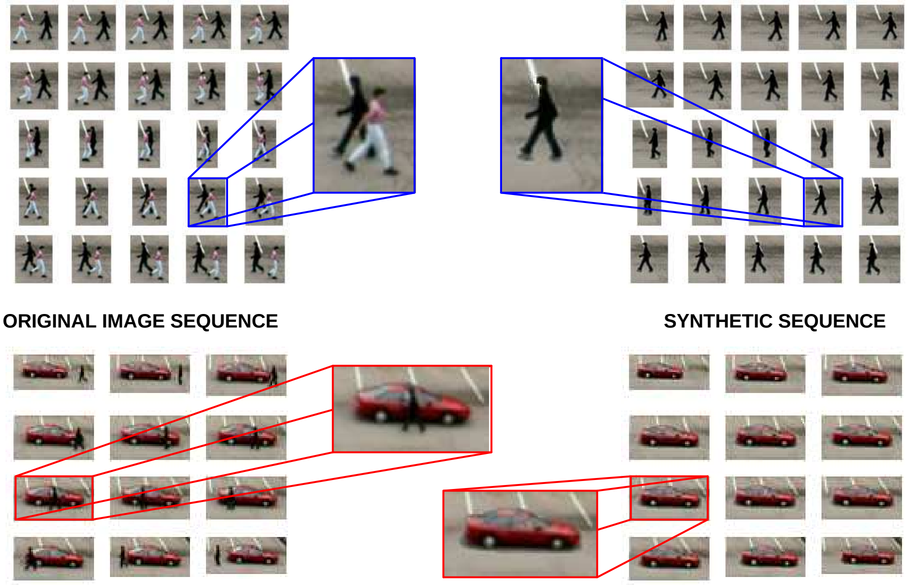


Figure 6. *Creating a synthetic character. The image sequences on the left are the originals. In both cases, the front-most character has been "killed" and a synthetic image sequence of the occluded character is generated based on its* periodic sequence*. These synthetic sequences are shown on the right.*

the position of the object is given as

$$x_n = x_{n-1} + T \times d_{[(n-1)\text{mod}P_N]} \tag{8}$$

where P_N is the size of $P(k)$. And the appearance of the object is $K \times \Omega_{[n\,\text{mod}P_N]}$ or $K \times F(\Omega_{[n\,\text{mod}P_N]})$.

7.1 Synthesising Occluded Characters

A special case of a synthetic character is needed when a "live" character is occluded by a "dead" one. In this case it is necessary to determine the frame in which the occlusion is about to occur and then create a *periodic sequence* for that character at that time. Once this is done, a synthetic character with scale factors $T = K = 1$ can be added to the *virtual video* stream for as long as the occlusion lasts. Determining occlusion is done using the *pixel contention* technique described in [16]. Figure 6 shows the results when a synthetic character is used to replace a real one which has been occluded.

8 Discussion and Conclusion

With the advent of advanced video understanding and real-time processing capabilities, the concept of *virtual video* is ripe for exploitation. The ability to interactively remove or insert characters from video streams has many applications in the entertainment industry, video conferencing, and virtual and augmented reality – just to name a few. Presented in this paper is an example application which shows how straightforward computer vision techniques can be married with computer graphics to create such an interactive video system.

Figure 7 shows some of the *virtual video* images generated by *Virtual Postman* in real-time running on an SGI O2 platform. The *Virtual Postman* game demonstrates some key ideas about *virtual video*. Adaptive, dynamic background subtraction provides an excellent method for extracting complete characters from a video stream, *and* producing a character-free background model which makes object removal an almost trivial task. Being able to track objects through video streams provides data by which objects can be analysed for rigidity and their periodicity determined. Using *periodic sequences* of templates and velocities allows cyclic motion to be captured and then replayed synthetically in arbitrary places and at arbitrary times. Putting these techniques together allows a player to interactively augment a video stream using data supplied exclusively from that stream.

References

[1] S. Adams. *The Joy of Work : Dilbert's Guide to Finding Happiness at the Expense of Your Co-Workers*. Harper-Collins, 1998.

[2] P. Anandan. A computational framework and an algorithm for the measurement of visual motion. *International Journal of Computer Vision*, 2:283–310, 1989.

[3] C. Brown and R. Nelson. Image understanding research at rochester. In *Proceedings of DARPA Image Understanding Workshop*, volume 1, pages 69–75, 1997.

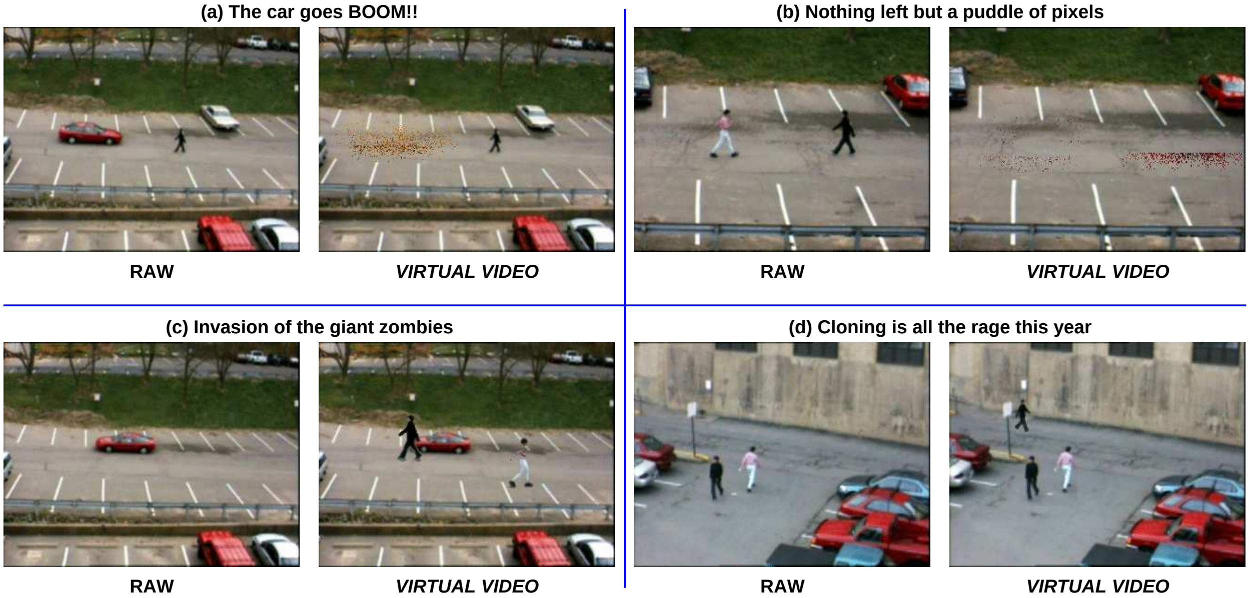


Figure 7. *Examples of* virtual video *images from* Virtual Postman. *(a) A car is "killed". The car is removed from the image and replaced by a computer graphic explosion. (b) Two characters are "killed", removed and replaced by puddles of pixels. (c) The same two characters of (b) come back later as giant zombies (at a spatial scale of* $K = 1.3$*) to terrorise a "real" car. (d) A "live" character is cloned as a zombie.*

[4] P. Burt, J. Bergen, R. Hingorani, R. Kolczynski, W. Lee, A. Leung, J. Lubin, and H. Shvaytser. Object tracking with a moving camera: An application of dynamic motion analys. In *IEEE Workshop on Motion*, 1989.

[5] H. Fujiyoshi and A. Lipton. Real-time human motion analysis by image skeletonization. In *Proceedings of IEEE WACV98*, pages 15–21, 1998.

[6] D. Gavrila. The visual analysis of human movement: a survey. *Computer Vision and Image Understanding*, 73(1):82–98, 1999.

[7] E. Grimson, C. Stauffer, R. Romano, L. Lee, P. Viola, and O. Faugeras. A forest of sensors: Using adaptive tracking to classify and monitor activities in a site. In *Proceedings of DARPA Image Understanding Workshop*, volume 1, pages 33–41, November 1998.

[8] M. Hansen, P. Anandan, K. Dana, G. van der Wal, and P. Burt. Real-time scene stabilization and mosaic construction. In *Proceedings of DARPA Image Understanding Workshopp*, 1994.

[9] I. Haritaoglu, L. S. Davis, and D. Harwood. w^4 who? when? where? what? a real time system for detecing and tracking people. In *FGR98 (submitted)*, 1998.

[10] M. Isard and A. Blake. Contour tracking by stochastic propagation of conditional density. In *Proceedings of European Conference on Computer Vision 96*, pages 343–356, 1996.

[11] A. M. Jazwinsky. *Stochastic Processes and Filtering Theory*. Academic Press, NY, 1970.

[12] T. Kanade, R. Collins, A. Lipton, P. Anandan, and P. Burt. Cooperative multisensor video surveillance. In *Proceedings of DARPA Image Understanding Workshop*, volume 1, pages 3–10, May 1997.

[13] T. Kanade, R. Collins, A. Lipton, P. Burt, and L. Wixson. Advances in cooperative multisensor video surveillance. In *Proceedings of DARPA Image Understanding Workshop*, volume 1, pages 3–24, November 1998.

[14] D. Koller, K. Daniilidis, and H. Nagel. Model-based object tracking in monocular image sequences of road traffic scenes. *International Journal of Computer Vision*, 10(3):257–281, 1993.

[15] A. Lipton, H. Fujiyoshi, and R. S. Patil. Moving target detection and classification from real-time video. In *Proceedings of IEEE WACV98*, pages 8–14, 1998.

[16] A. J. Lipton. Local application of optic flow to analyse rigid versus non-rigid motion. In *Submitted to International Conference on Computer Vision*, September 1999.

[17] S. M. Seitz and C. R. Dyer. View-invariant analysis of cyclic motion. *International Journal of Computer Vision*, 25(3):1–23, 1997.

[18] A. Selinger and L. Wixson. Classifying moving objects as rigid or non-rigid without correspondences. In *Proceedings of DARPA Image Understanding Workshop*, volume 1, pages 341–358, November 1998.

[19] P. Tsai, M. Shah, K. Ketter, and T. Kasparis. Cyclic motion detection for motion based recognition. *Pattern Recognition*, 27(12):1591–1603, 1994.

[20] C. Wren, A. Azarbayejani, T. Darrell, and A. Pentland. Pfinder: Real-time tracking of the human body. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(7):780–785, 1997.