Mobile Video Surveillance with Low-Bandwidth Low-Latency Video Streaming

Giovanni Gualdi
D.I.I. - University of Modena and Reggio Emilia
Modena, Italy
giovanni.gualdi@unimore.it

Andrea Prati
D.I.S.M.I. - University of Modena and Reggio Emilia
Reggio Emilia, Italy
andrea.prati@unimore.it

Rita Cucchiara
D.I.I. - University of Modena and Reggio Emilia
Modena, Italy
rita.cucchiara@unimore.it

ABSTRACT

This paper presents a system for remote live video surveillance. Videos are acquired from a fixed camera at 10 fps and QVGA resolution, compressed at 5 or 20 kbit/s with H.264, and streamed to a remote site, where they get processed by an automatic video surveillance system. The target surveillance application performs moving object segmentation and tracking. Both ends (video acquisition and processing) could be connected through a wireless network, specifically GPRS. The whole system is studied and optimized to maintain low latency. The reported experiments demonstrate that the proposed system is able to send up to four video streams over GPRS or E-GPRS network, without significantly affecting the performance of the automatic video surveillance system. Comparative tests have been performed with other existing streaming solutions.

Categories and Subject Descriptors

C.3 [**Special-Purpose and Application-Based Systems**]: Real-time and embedded systems; I.4 [**Image Processing and Computer Vision**]: Miscellaneous

General Terms

Design, Performance, Security

Keywords

Low-latency video streaming, Low-bandwidth video, H.264 video surveillance, Mobile and remote surveillance

1. INTRODUCTION

The huge diffusion of cameras in our cities and the urgent need of a safer everyday-life have contributed to push in few years the research in computer vision for video surveillance much farther than ever before. In addition, due to the technological advancements in distributed computing and wireless networks, video surveillance applications can now exploit ubiquitous cameras. They are not directly connected to local processing units, but transmit video frames to a remote server for intensive real-time analysis of the video. In the case of IP-based cameras exploiting large-bandwidth networks (such as Ethernet), remote video analysis might be similar to local processing, if the problems introduced by the network are negligible. However, this task becomes definitely more challenging when the camera is connected to the network through wireless low-capacity means, since a severe spatial and temporal compression of the video stream is mandatory, possibly almost nullifying the results of the automatic surveillance task.

Typically a process of automatic video surveillance requires to detect and track moving objects (such as people or vehicles). This process is very sensitive to noise and changing conditions and it is, thus, greatly influenced by the coding artifacts or the quantization introduced by the video compression. Consequently, the lower the bandwidth available, the greater the video compression, the more the noise affecting the correct video analysis.

More specifically, the typical video processing consists of (at least) three steps [12]: segmentation of moving objects, object tracking and object analysis. While the last step is very dependent on the goal of the application, segmentation and tracking processes are very general: a performance degradation in segmentation and tracking compromises the automatic surveillance system. The segmentation step is often based on background suppression techniques which compare the actual pixel values with the corresponding values in the (statistically) learned model of the static scene. It is evident that the frame compression can significantly affect this step by changing pixel values and making a sophisticated statistical background model useless. For this reason, in this paper we will measure the performance of the reported video streaming system in pixel-level moving object segmentation. The tracking step is also sensitive to frame skipping and to excessively-low framerates since it is typically based on object-to-track association on a frame basis and by considering a limited search area: thus, if an object moves too much on two successive frames, tracking is likely to lose it. For this reason, we will evaluate the ability of the system to track objects after a strong temporal compression due to video transmission.

In this paper, we describe a novel system for mobile wireless surveillance developed by merging an automatic surveillance system conceived for fixed wired cameras with a novel live streaming architecture for low-capacity networks. First, the paper describes our system for live streaming of video with low latency over low-capacity networks. Indeed, con-

MV'07, September 28, 2007, Augsburg, Bavaria, Germany.

sidering that the camera can be placed almost everywhere, a *network with* an (almost) *ubiquitous territorial coverage* is necessary; therefore, WiFi or UMTS can not be currently used due to their limited coverage; on the opposite, the GPRS network, thanks to the extremely wide geographical coverage of the GSM infrastructure, is optimal for the scope of the project; moreover, the GPRS-EDGE (Enhanced Data rates for GSM Evolution), also known as E-GPRS, version has been also used, given that it has almost the same coverage of GPRS but with a higher available bandwidth (theoretically between 160 and 236.8 kbit/s).

The overall streaming architecture has been described in [8], but here it is connected to a system, called Sakbot [6], for people video surveillance. The basic motivation of this work is to demonstrate that our streaming tool is more suitable to mobile video surveillance than commercial systems (such as Real Media or Windows Media) because it is able to fit in very limited bandwidth a video with sufficient frame quality and fluidity (few frame skipped) to almost preserve the performance of the video processing algorithms.

We tested the system in two different scenarios. The first is a platform for people surveillance at the campus in Modena, Italy, where several cameras have been used to control the main hall in the faculty building. The second is a system installed in a public park of Reggio Emilia, Italy, where people are segmented and tracked in a more cluttered environment [4]. The results of Sakbot system working with wired cameras are used for comparison with results achieved with wireless transmission of the videos over GPRS or E-GPRS.

2. MOBILE VIDEO SURVEILLANCE

In this work, we address mobile video surveillance with some operative constraints. They are not mandatory for the system to work, but they are adopted to make the comparison more general. Specifically, the whole system works in the following conditions:

- fixed camera: this allows to use background suppression techniques for object segmentation; moreover, video compression techniques are in general more efficient on fixed camera rather than on moving camera;
- the framerate is set to 10 fps; moreover, in order to have an effective object tracking, low frame-drop rates are required;
- low latency; video surveillance requires often quick reaction to changes in the scene (in order to allow interaction with the remotely observed scene); any latency greater than few (e.g. four) seconds is not acceptable for this kind of purposes.
- video bitrate compression is set to either 5 kbit/s or 20 kbit/s: in the first case, it is possible to send up to four video streams (possibly corresponding to four different cameras in a multi-camera surveillance system) on the average bandwidth of GPRS; however, this is only possible in case the video scene shows limited presence of moving objects (video motion covers less than a third of the scene) and rare illumination changes; if these hypotheses are not met, 20 kbit/s video compression has been used: in this condition it is possible to stream a single video over GPRS network, or four simultaneous videos over E-GPRS;
- the transmitted videos have resolution of QVGA - 320x240; this resolution is small but large enough to make precise segmentation in automatic surveillance;

The given constraints are very demanding: transmitting a video over 5 kbit/s or 20 kbit/s bandwidths, at QVGA resolution and with sufficient quality and framerate to be processed by a video surveillance system is far from being easy.

Being restricted to use low-capacity networks like GPRS, commercial compression techniques, such as MPEG-2 and MPEG-4 are not sufficient. Videos with CIF (352x288) or QVGA (320x240) resolution over a maximum bandwidth of 236.8 kbit/s, are achievable with MPEG-4 only by reducing either the frame rate (affecting video fluidity) or the image quality, and both these drawbacks will be in contrast with our conditions. For this reason, high interest has been shown in recent years on the evolution of MPEG-4 and H.263, the H.264/AVC [1] compression technology, that offers extremely good video quality even at very high compression rates: this makes it very suitable for the scope of our project. As a drawback it presents higher computational complexity if compared to MPEG-2 and MPEG-4 (both on encoder and decoder side), though it is currently possible to use it on real-time systems if the encoder is fed with reasonable size video streams and properly-tuned compression parameters [17].

Existing commercial or off-the-shelf streaming solutions are not sufficient to fit in our conditions. For example, Microsoft Windows Media® suite (Encoder, Streaming Server, Player) is a complete and powerful tool for video streaming, working with both stored and live videos. Unfortunately, since it has been developed more for entertainment purposes than for real-time live video streaming, this software introduces high latency and drops many frames. Even though this can not be a problem if you are watching a digital movie, it can instead affect the processing of the tracking algorithm. Moreover it cannot handle video streams at 5 kbit/s: the lowest it handles is 7 kbit/s with QQVGA (160x120) frame size. Other tools such as Helix Streaming Server® (Real Networks®) or Darwin Streaming Server® (Apple®) have been devised to handle multiple connections and severe computational load, more than to be used for low-latency live video streaming. The open source VideoLan, excellent in performance and flexibility, is not designed for low bandwidth video streaming: MPEG-TS network protocols is forced for video encapsulation, and 5 kbit/s cannot be achieved.

Given the high flexibility and performances of Helix Streaming solutions, we performed thorough comparative tests between our system and this platform. In the cases that Windows Media could sustain the bitrate, we also made tests with such platform.

3. RELATED WORKS

Video streaming has reached its peak of interest in the scientific community in the last ten years. A good survey paper on video streaming has been written by Lu in 2000 [15]. It reports of and discuss about the relevant signal processing issues related to video streaming and proposed possible solutions. Regarding video streaming, researchers have addressed the problem from several different perspectives. For example, one perspective addresses the allocation of com-

putational resources for a server with multiple requests [16]. Another set of papers focused on the system architecture, in terms of both models of communication (as in [5], where the analysis is based on RealNetworks® products, but not considering low-capacity networks) and data synchronization (as in the case of [9] where an inter-vehicle communication for live video streaming is considered, even though based on 802.11 WiFi networks and thus not suitable to our case). Some previous works have proposed systems for video streaming over low-capacity networks, such as GPRS. For instance, Lim *et al.*in [13] introduced a PDA-based live video streaming system on GPRS network. The system is based on MPEG-4 compression and contains several computational optimizations for working on a PDA. It can achieve a frame rate of 21 fps (frames per second) at the encoder side and 29 fps at the decoder side for transmitting at 128 kbit/s a video in QCIF (176x144) format. However, their system drops to 2-3 fps when transmission is over GPRS. This limitation is basically due to the use of MPEG-4. Moreover, no information on the latency of the system is provided. The work in [14], instead, solves the problem of transmitting realtime videos over GPRS by using frame skipping.

Mobile video surveillance has been envisioned in the literature as either classical video streaming with an extension over wireless networks, with no processing at remote side but only remote control by a human operator [18, 2, 3] or as a special case of distributed wireless sensor networks in which one type of the sensors corresponds to video sensors [11]. Moreover, most of these works do not address low-capacity networks. For instance Agrafiotis *et al.*[2] described the video transmission aspects of the European project WCAM (Wireless Cameras and Audio-Visual Seamless Networking) and made an excellent work in evaluating the performance (in terms of jitter buffer sizes, packet error rates, etc.) of H.264 using TCP/IP or UDP/IP stacks. However, this work is based on 802.11b/g networks. Cai *et al.*[3] reviewed video coding techniques for wireless networks, but also in this case the test-bed demonstration system is based on a 802.11b. One interesting example of video transmission over low-capacity networks and with the specification of low latency (or low delay) is reported in [18], but this work is focused on video streaming and not on successive video processing. Lam *et al.*presented a very interesting work [12] with a final objective similar to ours. However, in their case frame skipping is employed to fit in the low-bandwidth requirement with an intelligent filtering of frames. As mentioned above, this could complicate the tracking task and, moreover, requires to move part of the computational load on the local side of the camera. Additionally, results reported in [12] can fit in 5 kbit/s only with a very aggressive frame skipping. The approach proposed in this paper tends to the best trade-off between frame saving and image quality, by skipping very few frames and keeping image quality at a level sufficient for a correct segmentation of moving objects.

4. SYSTEM DESCRIPTION

The architecture of the system is sketched in Fig. 1. The server and the client applications are multi-threaded, and each block of the schema represents a main task to be performed and is processed by one dedicated thread. For the ease of future implementations, we decided to develop a C#/MS Visual .NET application that incorporates native C/C++ modules.

The video coming from the live surveillance camera is loaded by the server application which converts video data into a YUV (4:2:0) stream. This stream is converted in a H.264 stream by using the open source X.264 encoder. It should be mentioned that we modified the original X.264 source code by allowing it to load not only videos from file system, but also from a generic circular buffer, that allows higher flexibility, since it can be easily fed with any kind of source (file system, video device, video server, etc.). On the other side, the encoder thread asynchronously extracts frames from the buffer. Having asynchronous threads optimizes the processing since the execution of each thread is basically independent by the others. Moreover if the grabbing rate is higher than the encoding rate for short time, no video data will be lost. As drawback, the buffer introduces some latency; for this reason it is important to keep the buffer at low occupancy, therefore the grabbing frame rate should be set to be not higher than the average encoding frame rate. In standard conditions, with CPU load not too high, this is a reasonable assumption.

The raw H.264 encoded stream is seamlessly forwarded to the network streamer that packetizes it into UDP datagrams of fixed byte size. UDP is preferable with respect to TCP due to its prioritized dispatch over the network, but this comes at the cost of possible loss of packets. Nevertheless, thanks to the ARQ mechanism implemented on the Radio Link Control (RLC), our tests have demonstrated [8] that with our system over E-GPRS or even GPRS is very robust since an extremely low rate of packet are lost or received out of order. Moreover we preferred the use of raw UDP on RTP, since our system aims at delivering only a single video (image) data with no additional audio or text to be synchronized with.

At the receiver side, the client application extracts the UDP datagrams and merges them, rebuilding the H.264 stream. The H.264 decoder is based on `avcodec` and `avformat` libraries (from `ffmpeg`), and converts the stream back into a RGB stream that can be processed by the video processing subsystem (see Section 4.1).

In [8] we also report a detailed analysis to optimize the use of the UDP network buffer on the decoder side, in order to minimize datagrams losses and optimize playback fluidity. In the scenario of this paper, datagram losses avoidance is still essential, but, on the other hand, playback fluidity is not necessary to the video processing subsystem, that is therefore disabled.

4.1 Video processing subsystem

Moving people segmentation is achieved by using the background suppression approach called SAKBOT (Statistical And Knowledge-Based Object Tracker) presented in [6] and used in many different scenarios. The background pixels are defined by two models: the first statistical model updates the pixels at each frame using the temporal median function over the previous n sampled pixels; the second model exploits the knowledge of previous background and of the corresponding moving objects. Specifically, the pixels belonging to the current moving objects are not used for updating the model in order to prevent the gradual inclusion of slowly-moving objects into the background. Instead, pixels detected as foreground at previous steps but classified as noise or shadows by a suitable shadow detection algorithm are included in the statistical model. This approach

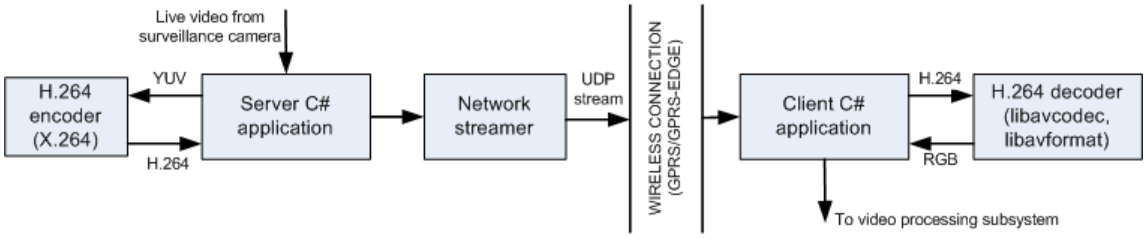


Figure 1: Architecture of the system.

is critical when a stopped object starts to move, generating two "foreground" objects, one real and one apparent (also called "ghost"). To avoid deadlock situations for the ghosts, a specific ghost suppression algorithm has been conceived. Further details can be found in [6, 4].

Moving objects detected by SAKBOT are then classified as person or non-person according with their geometrical shape and size (scaled according to their position on the ground plane). The objects detected as moving and validated as people are then tracked in each single view by means of an appearance-based algorithm. The algorithm uses a classical predict-update approach. It takes into account not only the status vector containing position and speed, but also the memory appearance model and the probabilistic mask of the shape [7]. The former, also called dynamic template in [10], is the adaptive update of each pixel in the color space. The latter is a mask whose values, ranging between 0 and 1, can be viewed as the probability for that pixel to belong to that object. These models are used to define a MAP (Maximum A Posteriori) classifier that searches the most probable position of each person in the scene. The tracking algorithm includes a specific module for coping with large and long-lasting occlusions. Occlusions are classified into three categories: self-occlusions (or apparent occlusions), object occlusions, and people occlusions. Occlusion handling is very robust and has been tested in many applications. It can keep the shape of the tracked objects very precisely and further details can be found in [7].

5. EXPERIMENTAL RESULTS

To test the performance of our system, we selected two quite different surveillance scenarios, one indoor taken at the hall of our building and one outdoor taken from a camera mounted in a public park. The first scenario is characterized by few moving people and no illumination changes, while the second is a less-controlled scenario, where several people move in the scene and illumination changes. The second scenario is obviously more challenging for both the video encoder and the automatic video surveillance system. For performance analysis we take one video for each of the scenarios, with a length of about 7 minutes each, and coded at 10 fps and a resolution of QVGA.

The accuracy of the overall system has been measured in terms of both pixel-level segmentation and object-level tracking, by comparing the results achieved by Sakbot on the original, non-compressed video with those obtained on the compressed, streamed video. Moreover, since video frames are lost during the transmission, we need a way to align the two videos (original and compressed) for having a correct comparison. Thus, the frame number (represented as binary code with squares black and white) was superimposed in a small corner of each frame at acquisition time. Then a simple image processing algorithm is used to automatically compute the frame number and obtain the exact frame-to-frame alignment between original and compressed.

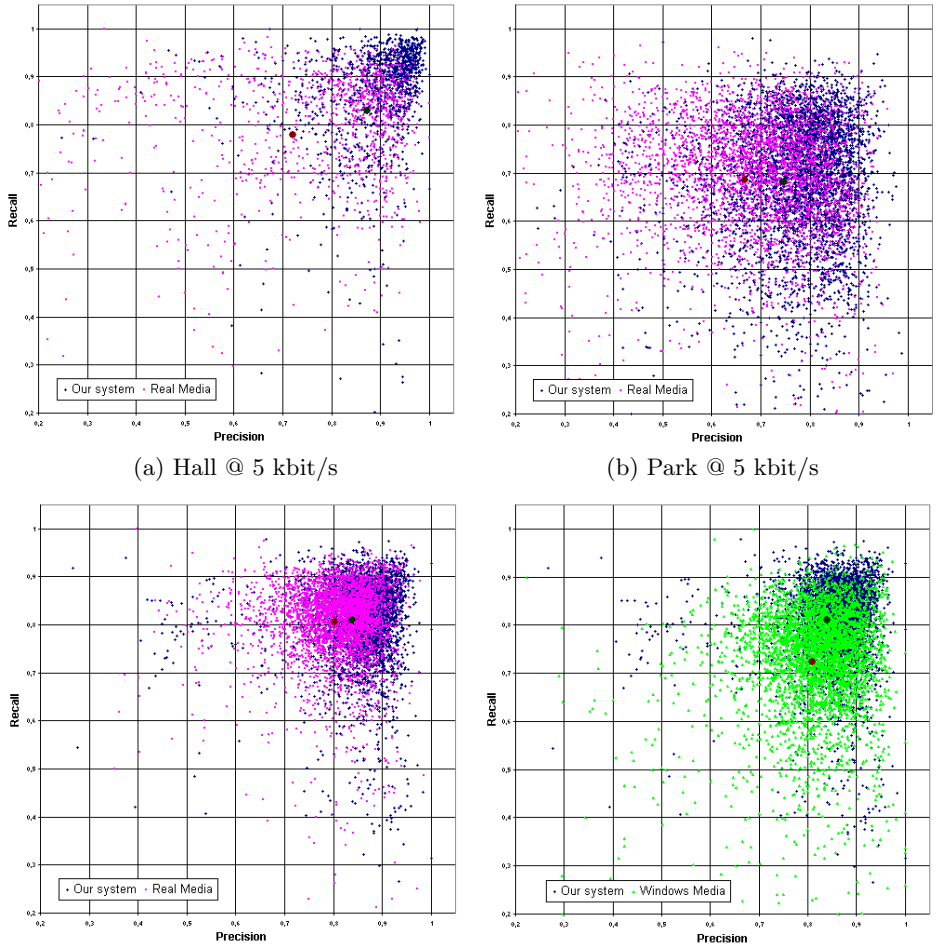
We considered the hardest case in terms of bandwidth, by supposing to send four video streams over a GPRS bandwidth, i.e. coding each video at 5 kbit/s. As a comparison, we tested both our streaming system and Real Media. Unfortunately, Windows Media and VideoLan were unable to code the video at such a low bandwidth. Since generating pixel-level ground truths for all the videos would have been quite a tedious and demanding task, as ground truth we used the segmentation generated on the original uncompressed video. In fact, our scope is to evaluate the loss in performance only due to the video compression and streaming and not the generic performance of the segmentation algorithm. Using this automatically-generated ground truth we compute, for each aligned frame, the recall R and precision P in pixel-level segmentation as follows:

$$R = \frac{TP}{TP + FN} \quad ; \quad P = \frac{TP}{TP + FP} \tag{1}$$

where TP indicates the true positives, FP the false positives and FN the false negatives. Plotting the recall and precision for each frame we obtained the graphs reported in Fig. 2. Obviously, if points move towards the upper-right corner (corresponding to $R = 1$ and $P = 1$) the accuracy is higher. The graphs also report the average recall and precision, represented by dark green (our system) and brown (Real and Windows) circles. The mean and variance of recall and precision are also summarized in Table 1. This table also shows the percentage of frame losses due to the strong compression rates. It is important to consider that means and variances were computed on the correctly received frames only.

The graph in Fig. 2(a) shows that, even with such a limited bandwidth, our system is very close to the segmentation obtained on the original video. For this reason, we decided not to report the results on 20 kbps. This situation does not hold in the case of the outdoor sequence (Fig. 2(b)): in this case the average recall of our system is less than 70% and the precision only about 75% (see Table 1). Thus, we also performed a test over E-GPRS using 20 kbit/s for each video. VideoLan does not support even this bitrate, but Windows Media does; thus, we report the comparisons between our and the two commercial systems in Fig. 2(c) and Fig. 2(d). Real Media shows a better recall on Windows Media, but have similar precision. However, our system outperforms them in terms of both recall and precision.

Regarding the object-level tracking, we manually compared the tracking results on original and compressed videos. Specifically we compute the ratio between the sum of the



(a) Hall @ 5 kbit/s

(b) Park @ 5 kbit/s

(c) Park @ 20 kbit/s - Real

(d) Park @ 20 kbit/s - Windows

Figure 2: Results recall vs precision for pixel-level segmentation. This figure is best viewed in the electronic paper version.

	Recall		Precision		lost
	mean	var.	mean	var.	frames
Hall @5 ours	82.9%	4.0%	87.3%	2.0%	0.4%
Hall @5 Real	77.9%	2.6%	72.0%	6.5%	8.8%
Park @5 ours	68.1%	2.2%	74.8%	2.9%	0.1%
Park @5 Real	68.7%	1.8%	66.8%	3.7%	14.6%
Park @20 ours	81.0%	1.0%	83.9%	2.0%	0.3%
Park @20 Real	80.5%	0.9%	80.2%	1.2%	0.0%
Park @20 Wind	72.4%	1.8%	80.9%	1.4%	1.8%

Table 1: Segmentation accuracy.

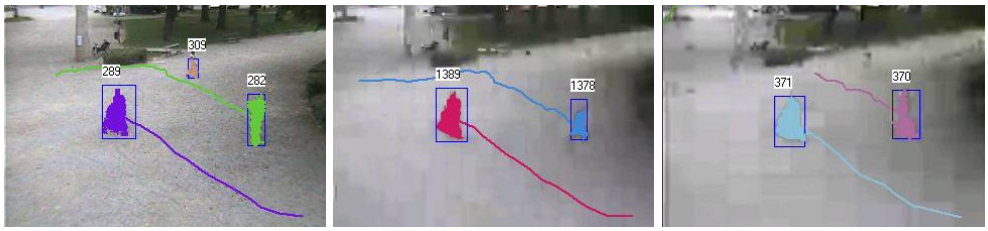
frames in which each object was tracked in the compressed video and the sum of the frames in which these objects were tracked in the original uncompressed video. Results are summarized in Table 2. A sample of the tracking in the park video sequence at 5 bit/s is shown in Figure 3: the tracking consistency is visually represented by the superimposed trajectory of the objects. Consider that the tracking numbers on the objects are just sequential IDs, thus do not need to be consistent between one video and the others. The latency of the complete system is made of two distinct components: the streaming (compression, network dispatch, decompression), and processing; the streaming delay is mainly due to the video complexity of the scene (that affects the compression time) and to the GPRS network. The average latency is approximately 1.7 s in our system, 3.8 s in Real Media and 6.4 s in Windows Media. Full details about streaming latency minimization are found in [8]. On the other hand the delay for the video processing is much lower, being always in the range of 0.1 to 0.2 s (when processed on a desktop PC).

	# objs	Accuracy		
		ours	Real	Wind.
Hall @5	29	96.51%	81.32%	n/a
Park @5	49	89.11%	67.95%	n/a
Park @20	49	91.91%	91.83%	89.87%

Table 2: Accuracy (in percentage) of the tracking.

6. CONCLUSIONS

This paper describes a fully working system for mobile video surveillance: video is acquired and, through strong



(a) (b) (c)

Figure 3: Snapshots showing the video tracking obtained with Sakbot on the Park sequence: (a) original, (b) our system @ 5 kbit/s, (c) Real Media @ 5 kbit/s.

compression and streaming over low-capacity mobile networks, dispatched live to a remote processing site, where automatic video surveillance is performed. Experiments give proof of the quality of the video surveillance in such conditions. The overall latency of our system is kept lower than 2 seconds, successfully addressing the requirements for live video surveillance. Our future works will be directed in experimenting higher complexity computer vision techniques on such strongly compressed videos.

This work was supported by project FREE SURF, funded by MIUR, and European Network of Excellence DELOS, sub-project Multimedia Interfaces for Mobile Applications.

7. REFERENCES

[1] Advanced video coding for generic audiovisual services. Technical report, ITU Rec. H624/ISO IEC 14996-10 AVC, 2003.

[2] D. Agrafiotis, T.-K. Chiew, P. Ferre, D. Bull, A. Nix, A. Doufexi, J. Chung-How, and D. Nicholson. Seamless wireless networking for video surveillance applications. In *Proceedings of SPIE - The International Society for Optical Engineering, 5685 (PART 1)*, pages 39–53, 2005.

[3] X. Cai, F. Ali, and E. Stipidis. Mpeg4 over local area mobile surveillance system. *IEE Colloquium (Digest)*, 3-10062:81–83, 2003.

[4] S. Calderara, A. Prati, and R. Cucchiara. HECOL: Homography and epipolar-based consistent labeling for outdoor park surveillance. *in press on Computer Vision and Image Understanding*, 2007.

[5] G. Conklin, G. Greenbaum, K. Lillevold, A. Lippman, and Y. Reznik. Video coding for streaming media delivery on the Internet. *IEEE Transactions on Circuits and Systems for Video Technology*, 11(3):269–281, Mar. 2001.

[6] R. Cucchiara, C. Grana, M. Piccardi, and A. Prati. Detecting moving objects, ghosts and shadows in video streams. *IEEE Trans. on PAMI*, 25(10):1337–1342, Oct. 2003.

[7] R. Cucchiara, C. Grana, G. Tardini, and R. Vezzani. Probabilistic people tracking for occlusion handling. In *Proc. of Int'l Conference on Pattern Recognition (ICPR 2004)*, volume 1, pages 132–135, Aug. 2004.

[8] G. Gualdi, R. Cucchiara, and A. Prati. Low-latency live video streaming over low-capacity network. In *Proc. of IEEE Int'l Symposium on Multimedia*, pages 449–456, 2006.

[9] M. Guo, M. Ammar, and E. Zegura. V3: a vehicle-to-vehicle live video streaming architecture. In *Proc. of IEEE Intl Conf on Pervasive Computing and Communications*, pages 171–180, 2005.

[10] I. Haritaoglu, D. Harwood, and L. Davis. W4: real-time surveillance of people and their activities. *IEEE Trans. on PAMI*, 22(8):809–830, Aug. 2000.

[11] Z. He. Resource allocation and performance analysis of wireless video sensors. *IEEE Transactions on Circuits and Systems for Video Technology*, 16(5):590–599, 2006.

[12] K.-Y. Lam and C. Chiu. The design of a wireless real-time visual surveillance system. *Multimedia Tools and Applications*, 33(2):175–199, 2007.

[13] K. Lim, D. Wu, S. Wu, R. Susanto, X. Lin, L. Jiang, R. Yu, F. Pan, Z. Li, S. Yao, G. Feng, and C. Ko. Video streaming on embedded devices through GPRS network. In *Proc. of IEEE Intl Conference on Multimedia and Expo*, volume 2, pages 169–172, 2003.

[14] Z. Liu and G. He. An embedded adaptive live video transmission system over GPRS/CDMA network. In *Proc. of Intl Conf. on Embedded Software and Systems*, 2005.

[15] J. Lu. Signal processing for internet video streaming - a review. In *Proc. of Conf on Image and video communications and processing*, pages 246–259, 2000.

[16] M.-T. Lu, C.-K. Lin, J. Yao, and H. Chen. Complexity-aware live streaming system. In *Proc. of IEEE Int'l Conference on Image Processing*, volume 1, pages 193–196, 2005.

[17] A. Puri, X. Chen, and A. Luthra. Video coding using the H.264/MPEG-4 AVC compression standard. *Signal Processing: Image Communication*, 19:793–849, 2004.

[18] C.-F. Wong, W.-L. Fung, C.-F. Tang, and S.-H. Chan. Tcp streaming for low-delay wireless video. In *Second International Conference on Quality of Service in Heterogeneous Wired/Wireless Networks*, 2005.