Testing performance of current video codecs in teleoperated mobile robot applications: a practical experience

Pablo Piñol, Otoniel López, Miguel Martínez-Rach,
M.P. Malumbres, José Oliver and Carlos Calafate
Universidad Miguel Hernández de Elche
Spain

1. Introduction

Mobile robots are used in a wide variety of applications and environments. We have a lot of examples like spatial missions, planet surveying, industrial cleaning, people transport, surveillance and security, material storage, underwater tasks, bomb deactivation, tunnel works, building industry, leisure toys, …

One of the problems a mobile robot faces to is "knowing what is around". To solve this problem the robot uses perception. There are many different types of "perceptions" of the surrounding that a robot can get using sonar, laser, infrared, pressure sensors, temperature measurements, sound analysis and of course real-time video. Real-time video can be used by the robot itself, if we have an autonomous mobile robot or by an operator if we have a teleoperated robot.

Real-time video produces a huge amount of data which requires a great deal of resources of storaging capacity (for saving video data) and network bandwidth (for transmission). But, even though storage capacity and network bandwidth keep growing in present-day devices, video compression is mandatory.

When autonomous robots have to communicate with each other or when a teleoperator needs feedback of the environment or wants to send navigation commands to the robot we use a wireless link. In the first stages of mobile robot communication, proprietary RF systems were used. Nowadays WLANs are a good choice for implementing communications in a mobile robot due to its good characteristics: standardization, low cost, flexibility, high bandwidth, a certain degree of security, …

IEEE's 802.11 (IEEE-WG-802.11, 1999) standard is being increasingly used throughout corporations worldwide due to its good balance of cost, range, bandwidth and flexibility and along with 802.11e (IEEE-WG-802.11, 2002), that supplies QoS at MAC layer, form a good base to support multimedia traffic.

Although the main application area of wireless networks is related with user access to existing wired network infrastructure, other applications, like telerobotics can benefit from WLANs technology. The development of teleoperated systems has gained considerable

attention due to the new potential applications, such as remote production monitoring (Luo et al., 1999), remote exploration and manipulation in inhospitable environments (Hirzinger et al., 1993), tele-surgery (Kwon et al., 1999), and remote training. Important issues concerning communication channels, random propagation delays, bandwidth limitations, fault-tolerance, synchronization, tele-presence, and the stability of the robotic systems involving human operators have all been taken into account in different works across the literature (Brady & Tarn, 1998); (Elhajj et al., 2001). Most of them consider Internet as the interconnection network between telecontrolled systems and control stations (Goldberg & Siegwart, 2002).

The use of wireless channels for teleoperating mobile robots is not new. Most of the existing wireless-controlled robots use a specific (non-standard) radio-modem for communications between control station and robot. But this wireless channel is usually a point-to-point dedicated wireless link which works under good signal quality environments. Therefore, in that case, the wireless link does not represent a drawback in terms of performance of the overall system.

Other works describe implementations of teleoperation systems with standard wireless devices. In (Fong et al., 2001) authors use a PDA with an IEEE 802.11 wireless interface for telecontrolling a robot, and in (Sgouros & Gerogiannakis, 2003) the same was done through a WAP connection. However, they do not analyze the behavior and performance of wireless link for the correct operation of this kind of applications.

In IEEE 802.11 networks working in infrastructure mode, the access points (base stations) provide the network connectivity between mobile and wired hosts. WLAN nodes share the medium through a CSMA-CA protocol, so network latency may be significant and dependent of current traffic load. The time-varying conditions of the wireless channel, roaming processes and certain node mobility patterns, may disturb established communications, increasing packet loss ratio and the average delay and jitter, up to intolerant levels for certain kind of multimedia applications.

On the other hand, under this kind of error-prone environments, delivery of encoded video is a challenging task if a minimum QoS is demanded from applications. Video encoders should be prepared to recover damaged bitstreams employing one or several error resilience tools. Video codecs will play an important role in the overall application performance, being critical the ability of concealing network delivery errors to assure the minimum required video quality.

In this chapter we will show the results of studying three present-day codecs (H.264/AVC, MPEG-4 (DivX), VP7) and how they manage with packet losses (due to node mobility and time-varying signal quality) in a real wireless environment. A prototype has been built consisting of a mobile robot with a laptop on it. This laptop is equipped with a webcam and a wifi network card. Both timing measurements and video quality assesment have been performed in several scenarios. Our experiments have been done at different rates of background traffic and with the robot moving from good signal areas to places where wireless signal gets lost. We have used different packet sizes, different codec modes (*intra*/*inter*) and different video frame sizes (QCIF/CIF/D1). We have employed DirectShow technology and a client/server scheme. Some simulation has also been done to tune up our prototype.

The chapter is organized as follows: in section 2 we describe the teleoperated framework that will be used in this work. Section 3 shows a detailed study of end-to-end delays in both

video and navigation data flows. In section 4 we analyze commercial video codecs and how they recover damaged bitstreams in scenarios where packets are lost. In section 5 we devise some real scenarios to measure the impact of different network conditions in our telecontrolled robot system. Finally, in section 6, some conclusions are drawn.

2. Mobile Robot System

In Fig. 1 we show the client/server framework implemented for this study. The server hardware consists of an ER1 mobile robot from Evolution Robotics (ER1, 2009) which holds a laptop on it (see Fig. 2(a)). The laptop has a webcam for capturing a video stream and a 802.11 WLAN network adapter which sends the video stream to the client (control station) and receives navigation orders to operate the robot. The robot hardware is controlled by the laptop through a USB port. The client hardware consists of another laptop connected to a Fast Ethernet network. We also have two workstations for injecting background traffic. The traffic sender is connected via a Fast Ethernet link and the traffic receiver is connected via a WLAN link.

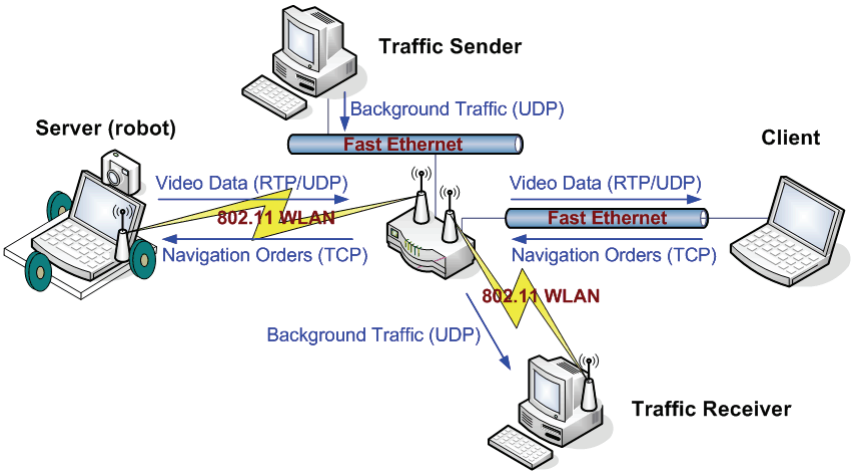
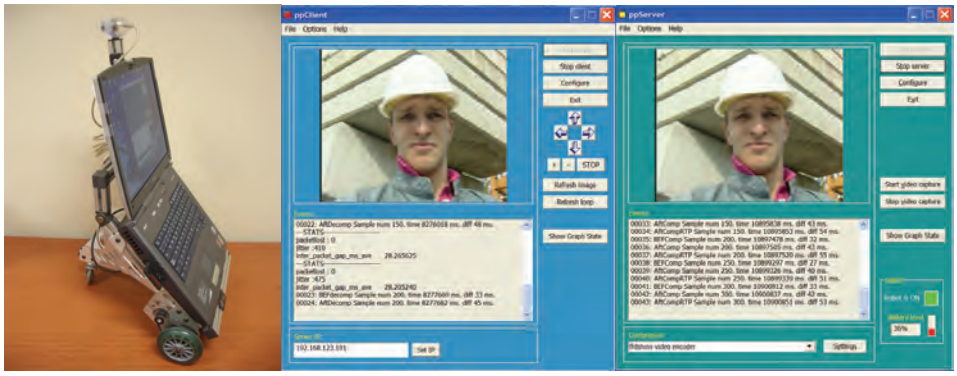


Fig. 1. Overview of the telerobotic experimental platform.

The software applications developed (see Fig. 2(b,c)) are based on DirectX/DirectShow architecture (DirectShow, 2009). This enables us to use any of the commercial codecs available for this technology. DirectShow applications handle data through different "boxes" called filters which process and transform information. So, for encoding and decoding a video frame we will use a codec filter. Let's talk about data flows in our framework. There are two different kind of data flows between client and server: video stream and navigation commands.

Video stream consists of RTP/UDP (Real-time Transport Protocol) (RFC-3550, 2003) video packets which are sent by the server and collected by the client. First, the laptop in the robot captures a frame using the webcam. This frame is compressed using one of the codecs installed in the laptop (we can also select "uncompressed" if we do not want to apply

compression). After this is done, the RTP filter, which is based on LIVE555 Streaming Media Library (Live Networks, 2009) packetizes the compressed frame and then sends the packets to the client through the IEEE 802.11 link. The client receives the packets, arranges them and reassembles the compressed frame. This frame is decoded and finally rendered on the user interface window.



a b c

Fig. 2. (a) ER1 and laptop, (b) Client application, (c) Server application.

The other data flow, navigation commands, is sent by the user in order to control the robot. When the user issues a navigation command using the GUI client software, a short TCP packet is sent to the server. After receiving the navigation command, the server processes it in order to control the robot motion subsystem.

3. Delay Analysis

In this section we are going to analyze the control loop delay sources in both video and navigation flows. For that purpose we have selected three well known video test sequences: *Foreman* video sequence (400 frames) at QCIF format (176x144 pixels), *Coastguard* video sequence (300 frames) at CIF format (352x288 pixels) and *Mobile* video sequence (320 frames) at D1 format (720x576 pixels). For all video sequences we have fixed a common frame rate of 30 fps and a 4:1:1 YUV color space (12 bits/pixel).

We have established three different compression levels: (1) the lowest image quality (a low-but-acceptable quality, around 27 dB of PSNR – Peak Signal-to-Noise Ratio), (2) a medium video quality (around 33 dB), and (3) the highest image quality, offering around 39 dB (where further improvements on the quality are nearly imperceptible).

We have chosen three DirectX/DirectShow codecs under evaluation: MPEG-4 (MPEG-4, 2009), H.264/AVC (H.264, 2009) and VP7 (VP7, 2009). There are two available coding modes: *intra* and *inter*. In *intra* coding mode every video frame is individually encoded. However, when *inter* coding mode is configured, some video frames are individually encoded (key frames) and the rest are encoded taking as reference other frames. For *inter* mode we have configured them to produce a key frame every 15 frames (so we label it as *inter-15*).

3.1 Video stream delay

Video stream delay is the time elapsed since the robot's webcam captures a single frame until the client renders it to the user. The first source of delay is generated by the robot's webcam. By *vd1* we represent the time the webcam takes to capture a video frame. From the webcam specifications, it takes from 5 to 20 ms to capture an image. Our webcam has resolutions that range from 160x120 to 640x480 pixels. So, we have lineally determined the capturing delay for each video format.

When compressing video, we will encounter delay *vd2* which measures the time the video encoder takes to compress one video frame. This delay will depend on different variables such as the encoder used, the requested bitrate (opposite to the reconstructed video quality) and other compression parameters. If we do not compress video then *vd2* will have a value of 0. In Table 1 we show the average delay for encoding and decoding a single video frame using a medium level of compression. *Intra* mode is usually faster than *inter* mode but produces a greater number of packets for delivery.

By *vd3* we represent the time required to perform the packetization and delivery of one frame. We have used Wireshark (Lamping et al., 2006) network analyzer for measuring network delays. In Table 2 we show this delay using a packet size of 1272 bytes and a medium level of compression. We have measured it without background traffic (no traffic in the IEEE 802.11 and Fast Ethernet segments). We have also measured uncompressed video delivery at a resolution of 160x120 pixels. Here, the average delivery delay is 24.68 ms (with 23 packets per frame).

		QCIF - Foreman		CIF - Coastguard		D1 - Mobile	
		Encode	Decode	Encode	Decode	Encode	Decode
H.264	intra	7.48	3.47	27.78	13.13	105.01	42.20
	inter-15	7.60	1.80	35.36	8.20	113.31	17.82
MPEG-4	intra	0.95	1.42	2.73	8.04	9.79	14.90
	inter-15	2.68	0.96	10.43	2.35	36.50	7.81
VP7	intra	14.61	2.43	26.07	8.66	74.53	58.50
	inter-15	24.12	5.50	23.32	14.92	35.98	26.69

Table 1. Encoding and decoding delay (ms)

		QCIF - Foreman		CIF - Coastguard		D1 - Mobile	
		pkt/fr	ms/fr	pkt/fr	ms/fr	pkt/fr	ms/fr
H.264	intra	2.25	0.88	8.94	8.63	28.08	29.70
	inter-15	1.09	0.65	3.75	2.36	5.06	3.91
MPEG-4	intra	2.27	0.80	8.92	7.74	26.91	27.11
	inter-15	1.08	0.51	3.53	2.32	5.08	3.59
VP7	intra	2.22	0.85	8.88	8.25	27.97	29.64
	inter-15	1.07	0.56	3.45	2.30	4.96	3.29

Table 2. Delivery delay (ms) per frame and packets per frame

Once a packetized frame has been assembled at the client, it is delivered to the uncompressing module to get the reconstructed version of the original frame. The time required by the decoder to reconstruct the original frame is represented by *vd4*. If we do not compress video then *vd4* is 0. See Table 1 for average decoding delay.

The overall video flow delay will be the sum of all these delays (*vd1* + *vd2* + *vd3* + *vd4*). At a same level of compression (similar bitrate) all codecs produce a similar delay (*vd3*) when delivering a single frame through the network. So the difference between the codecs will be determined by the time needed to encode and decode one frame (*vd2* + *vd4*). MPEG-4 has always proved to be faster than the other two codecs (Fig. 3).

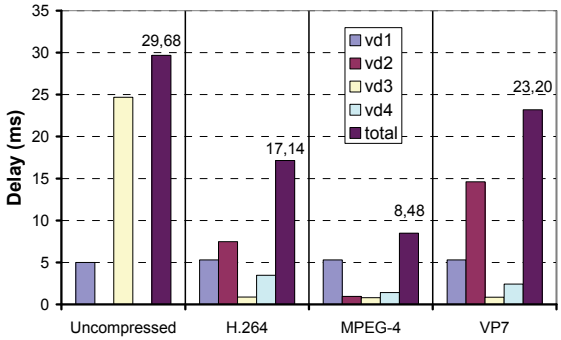


Fig. 3. Video stream delay for different codecs (QCIF resolution, *intra* mode).

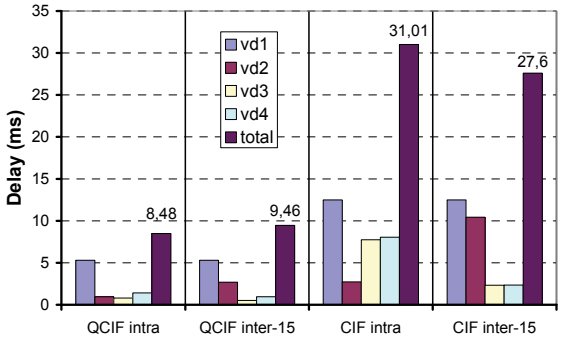


Fig. 4. Video stream delay for different resolutions and modes (MPEG-4 codec).

We have noticed that coding-decoding delay in *inter-15* mode is usually higher than in *intra* mode. At low resolution images (QCIF), the number of packets generated by both methods is very low. So, in the overall video flow delay, using *intra* mode is faster (Fig. 4). However, for medium and large resolution images (CIF, D1) the number of packets generated with *intra* mode is significantly greater than *inter-15*. So, in terms of overall video delay, *inter-15* mode is faster than *intra* mode.

3.2 Navigation commands delay

The other flow of data carries navigation commands from the client to the server in order to teleoperate the robot. The available navigation commands are the following: move forward/backward, turn left/right, increase/decrease speed, and stop. We send navigation

commands in very small TCP packets. We use TCP protocol instead of UDP protocol because it guarantees reliable and in-order delivery of navigation commands.
The first delay we find corresponds with the time elapsed since the user issues a navigation command until the server receives it. We label this delay as *nd1*. When the server receives a navigation command it immediately sends the order to the robot hardware. By *nd2* we indicate the time elapsed since the server sends the order until the robot begins to execute it (and it can proceed to wait for another command). We have measured the average time for executing one navigation command, resulting in the following delays: 22.3, 12.2, and 18.3 ms required by forward/backward, turn and stop commands, respectively.

3.3 Overall delay

The overall delay indicates the time the prototype takes for a complete teleoperation action (control loop). It will depend on several factors: frame size, codec used, compression level, navigation command, network status, etc. For example, if we send a forward moving command using an MPEG-4 codec in *intra* mode, a medium level of compression and QCIF video format, we will have an overall delay of 30.78 ms (video delay + navigation command delay). The same configuration but with CIF video format will give an overall delay of 53.31 and 49.9 ms for *intra* and *inter-15* coding modes, respectively.

4. Simulation Tests

Now, we will analyze the video codec behavior under a scenario with and without packet losses. Bit error patterns are not considered here since IEEE 802.11 network technology uses ACKs for every packet, so the only delivery error is packet loss. In the scenario represented in Fig. 1, the access point (US Robotics USR8054) uses an MTU of 1300 bytes (when working in turbo mode) so removing the IP and UDP headers, we will have a maximum RTP packet size of 1272 bytes. We have done our tests with a pair of packet sizes, 1272 and 512 bytes (512 bytes is the default payload size used by some traffic generators). RTP library can be configured to randomly drop packets with a specific PLR (Packet Loss Ratio), so we have used this feature to simulate an error prone environment.

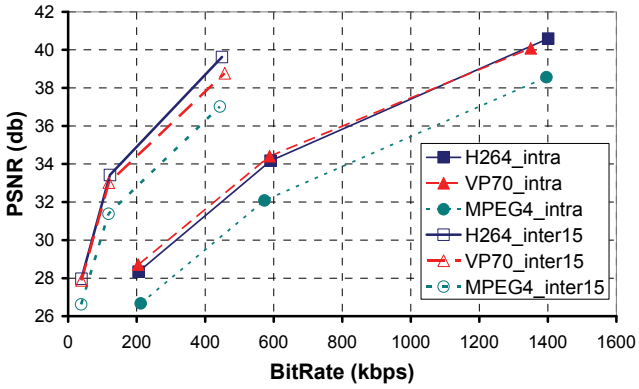


Fig. 5. R/D performance: objective PSNR quality comparison.

Fig. 6. R/D performance: subjective quality at lowest bitrate.

In Fig. 5 we show the R/D (Rate/Distortion) performance of the evaluated codecs with Foreman (QCIF) video sequence and no packet losses. This figure shows what we obtained in most of our loss-free tests: at similar bitrates H.264/AVC offers the best quality (nearly followed by VP7) and MPEG-4 offers the worst quality with difference. In Fig. 6 we can perceive the subjective quality difference between the codecs for the first frame of the video sequence, compressed at the lowest bitrate.

4.1 Simulated packet losses (evaluation)

In Fig. 7 we evaluate the behavior of the selected video codecs when the video delivery system begins to lose packets. We have used Foreman (QCIF) video sequence working in *intra* mode, a medium compression level, and a fixed packet size of 1272 bytes. We show the R/D results with different PLR values. It can be seen that in absence of packet losses H.264/AVC and VP7 get similar results, much better than the ones obtained by MPEG-4. It can also be seen that H.264/AVC does not fall as much as VP7 when PLR increases.

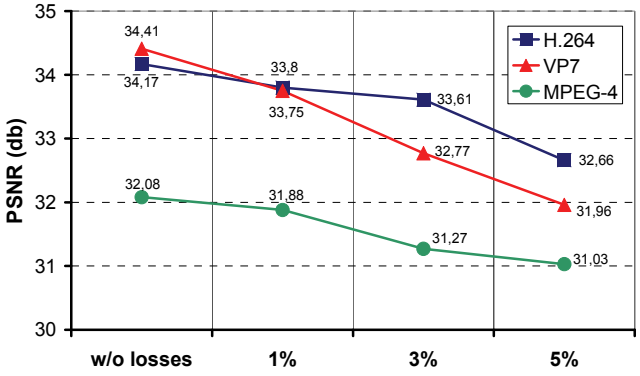


Fig. 7. R/D performance of codecs under packet losses.

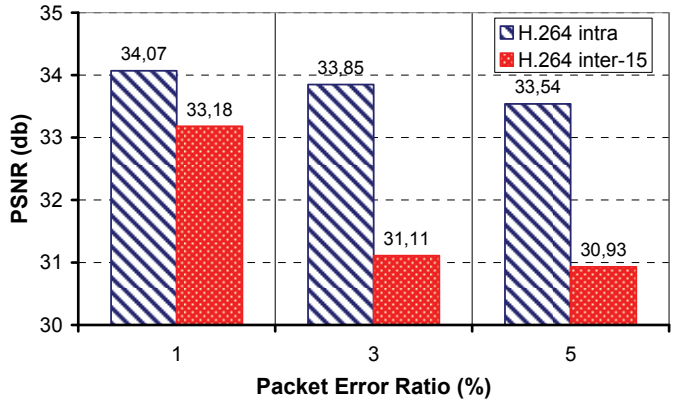


Fig. 8. Resilience of *intra* and *inter* compression modes.

Fig. 8 shows the video quality measures for Foreman (QCIF) video sequence using H.264/AVC codec with a medium compression level and a packet size of 512 bytes. Here, we show the difference between *intra* and *inter* modes in terms of error resilience. The main reason that explains the lower error resilience behavior of *inter* mode coding is due to the propagation of errors to several frames.
So, if we find an error when decoding one frame, this error will propagate to those frames that require the former one to be reconstructed. As a consequence, the reconstructed video quality will be significantly reduced.

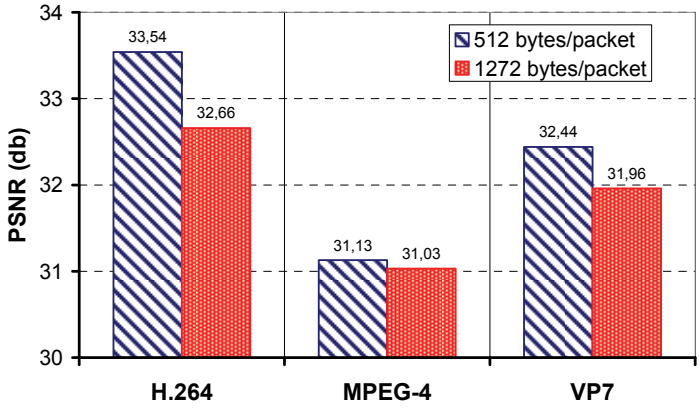


Fig. 9. R/D comparison for two packet sizes (*intra* mode, PLR=5%).

In our tests we realized that, at a fixed PLR, the use of big packet sizes introduces more distortion than using small packet sizes (Fig. 9). But, as we will see later, at real scenarios, the use of small packets produces more packets to send and therefore there are more probabilities for them to get lost, so small packets provide a higher PLR and, as a consequence, a worse video quality.

The most outstanding outcomes for the tests we have performed are that, in general, at the same bitrates H.264/AVC and VP7 offer better quality than MPEG-4 (remember that in section 3 we observed that MPEG-4 was the faster codec) and *intra* mode is more error resilient than *inter* mode.

5. Real Scenarios Tests

We have performed two test sets under real scenarios. The first one was in an environment with no background traffic, and the second one was in an environment with different levels of background traffic load. All the tests have been done indoor with the robot moving forward at a constant speed of 55 cm/s. It starts moving at the beginning of a corridor nearby the access point (where it has 100% signal strength) and goes away during 110 seconds up to the end of the corridor, where the signal gets lost (see Fig. 10). Traffic Sender and Traffic Receiver are both static and near the access point. We have used Coastguard (CIF) video sequence concatenated ten times (so it has a length of 3000 frames and is 100 seconds long) with a frame rate of 30 fps. We have setup the codecs to provide a bitrate of 2 Mbps in *intra* mode.

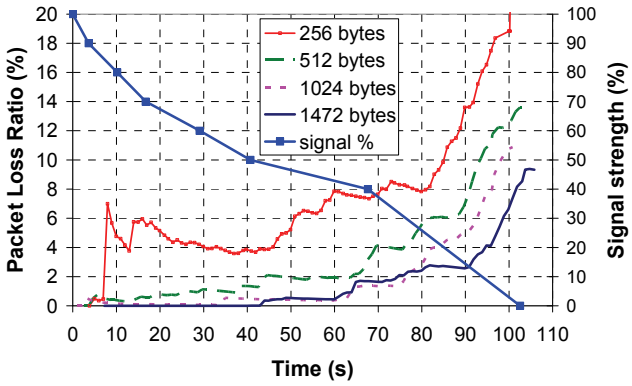


Fig. 10. Signal strength and PLR for different packet sizes.

First, we have done some tests with MPEG-4 codec to evaluate the system performance for different packet sizes without background traffic (unloaded network). In Fig. 10, the packet size that gets the lowest PLR is 1472 bytes (maximum size for an Ethernet MTU of 1500 bytes). So, from now on, we will fix the packet size to 1472 bytes.

The second set of tests was done introducing different background traffic loads in the system network. We have used D-ITG (D-ITG, 2009) traffic generator and D-ITG GUI (D-ITG-GUI, 2009) graphical interface in both sender and receiver to inject background traffic (CBR UDP stream) to our WLAN. For these tests we have used all codecs in *intra* mode with a packet size of 1472 bytes (as mentioned above). We have injected seven different background traffic loads: from 2048 to 3277 Kbps. Traffic loads under 2048 Kbps have a negligible incidence on the video flow delivery and those over 3277 Kbps drive the network to a saturation state.

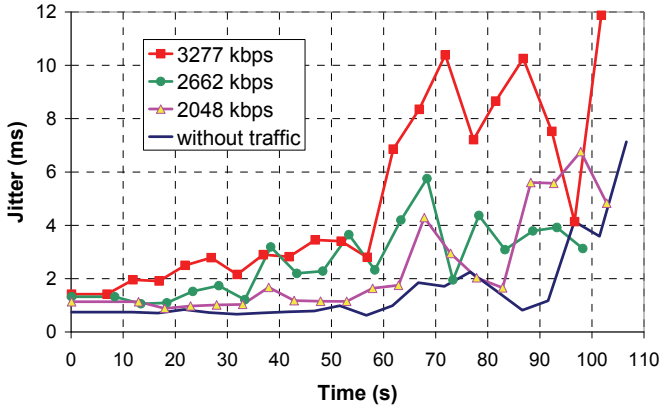


Fig. 11. Jitter comparison at different background traffic loads (MPEG-4 video codec).

In Fig. 11, a comparison of experienced jitter (average packet delay variance) during the session is shown under different background traffic loads. The observed jitter shows a different behavior with respect background traffic (network contention) and received signal strength.

When the signal strength is above 50%, the jitter fluctuations are mainly due to network contention (background traffic). However, when signal strength is below 50%, the jitter wildly changes, amplifying the effect that traffic background has over it.

In Fig. 12, we show the time-average PLR in a session where MPEG-4 was used with different background traffic loads (as reference we show the results without background traffic). As it can be seen, packet losses are directly influenced by signal strength and traffic load for all the region of coverage, in a similar fashion than in Fig. 10.

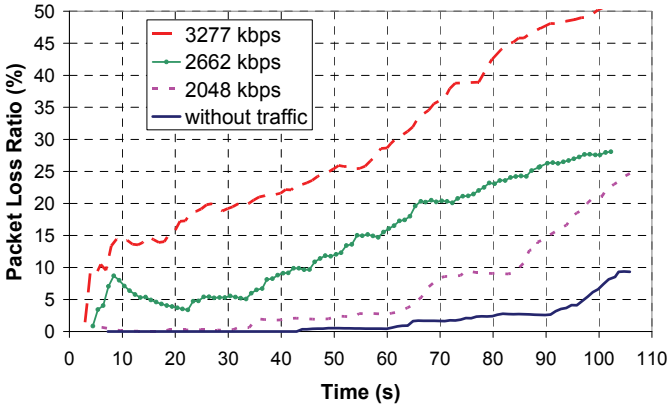


Fig. 12. PLR evolution using MPEG-4 codec with different background traffic loads.

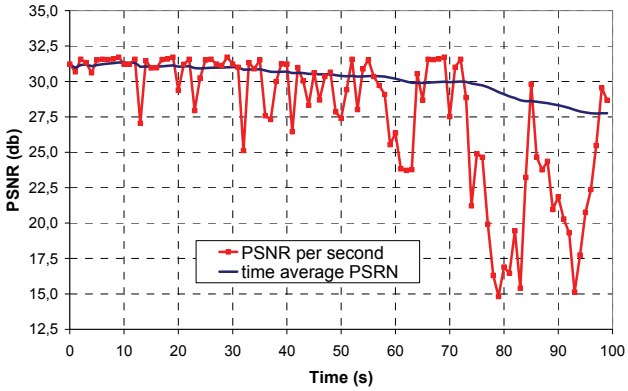


Fig. 13. Quality of MPEG-4 reconstructed video with a background traffic load of 2259 Kbps.

In Fig. 13 we show the reconstructed video quality of the received video sequence using the MPEG-4 codec and a background traffic load of 2259 Kbps. We show the evolution of PSNR (each point represents the average PSNR of 30 frames) and the time-average PSNR. It can be seen that when a packet or a burst of packets is lost the PSNR falls drastically (we have found a PSNR decay of up to 15 dB for a single frame). This situation appears in those areas where the received signal strength is under 50%. In those situations there are a lot of lost packets, typically in bursty fashion. So, in most cases, the frame can not be reconstructed (the frame is declared as lost) or, if the codec has enough information for decoding the frame, the quality of the frame recovered is too low.
In Fig. 14 we compare the selected codecs and the video quality they provide with two different traffic loads. As it can be seen, the H.264/AVC video codec always obtains better results than the other two, showing the same behavior than the one observed without packet losses. However, VP7 shows lower performance than MPEG-4, since this codec is not able to reconstruct a frame when the first part of it is missing. So, to avoid VP7 crashing, we drop the incomplete frame at RTP receiver process. This is the main reason of VP7 low performance.

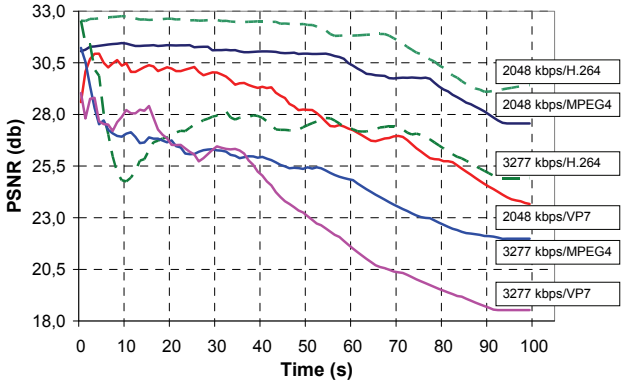


Fig. 14. Quality of reconstructed video with different codecs and traffic loads.

In Fig. 15 we can see the subjective quality of a reconstructed video. Fig. 15(a) shows frame number 2191 of the video sequence without any compression. This corresponds to second 73 of the sequence. In Fig. 15(b) we can see the same frame after it has been encoded and decoded using VP7 codec (without any loss). For this frame we get a value of 31.47 dB of PSNR. In Fig. 15(c) we can see the same frame after reconstruction in a test where packet loss has occurred. This frame offers a value of 14.50 dB of PSNR.

Fig. 15. Subjective quality: (a) Coastguard original frame number 2191, (b) VP7 encoded-decoded, (c) VP7 reconstruction with packet losses.

6. Conclusions

In this chapter we have analyzed the performance of current commercial video codecs for video streaming through 802.11 networks running in a remotely teleoperated mobile robots testbed. We have developed a client-server application based in DirectX/DirectShow architecture for testing video compression, data delivery and error resilience behavior in a real scenario.

A first analysis was performed to study the teleoperation control loop delays and the performance of video delivery process in order to get a first impression of overall system behavior. For this kind of applications, compression is mandatory for proper operation in order to cope with the minimum video quality and bounded delay demanded by them. The use of video compressors allows us to adjust the required quality without exhausting network resources like available bandwidth and they significantly reduce the control loop delay improving the functionality of this kind of applications.

With respect to packet losses we have observed that H.264/AVC codec is the one that best performance results achieves. Also, we have checked that *intra* coding mode gets better results than *inter* mode, since it avoids error propagation. In the packetization process, the RTP packet size determines the resulting application bandwidth (goodput). Results show that the longer the packet size is, the higher goodput the application gets.

Finally, we think that telerobotic applications running under standard wireless network technologies like IEEE 802.11 can benefit from the use of state-of-the-art video encoders, like H.264/AVC, to improve throughput, error resilience and real-time feedback.

7. References

Brady, K. & Tarn, T.J. (1998). Internet-Based Remote Teleoperation, *Proceedings of the IEEE Conference on Robotics and Automation*, pp. 65-70, 0-7803-4300-X, Leuven, Belgium, May-1998

DirectShow. (2009). Microsoft Developer Network, DirectShow Application Programming Interface , *http://msdn.microsoft.com/en-us/library/dd375454(VS.85).aspx*

D-ITG. (2009). Universita' degli Studi di Napoli "Federico II" (Italy), Distributed Internet Traffic Generator, *http://www.grid.unina.it/software/ITG*

D-ITG-GUI. (2009). Volker Semken, Graphical User Interface for D-ITG, *http://www.semken.com/projekte*

Elhajj, I.; Xi, N.; Fung, W.K.; Liu, Y. H.; Li, W.J.; Kaga T. & Fukuda T. (2001). Supermedia in Internet-Based Telerobotic Operations, *Proceedings of the 4th IFIP/IEEE International Conference on Management of Multimedia Networks and Services*, pp. 359-372, 3-540-42786-4, Chicago, IL, USA, November-2001

ER1. (2009) Evolution Robotics Personal Robot System, *http://www.evolution.com/er1*

Fong, T.W.; Conti, F.; Grange, S. & Baur, C. (2001). Novel interfaces for remote driving: gesture, haptic and PDA. *Proceedings of SPIE Mobile Robots XV and Telemanipulator and Telepresence Technologies VII*, pp. 300-311, 0-8194-3860-X, Boston, MA, USA, November-2001

Goldberg, K. & Siegwart, R. (2002). *Beyond webcams: An introduction to online robots*, MIT Press, 0-2620-7225-4, 2002

H.264. (2009). H.264/MPEG-4 AVC free software library, *http://x264.nl*

Hirzinger, G.; Brunner, B.; Dietrich, J. & Heindl, J. (1993). Sensor-Based Space Robotics-ROTEX and Its Telerobotic Features. *IEEE Transactions on Robotics and Automation*, Vol. 9, No. 5, October-1993, pp. 649-663, 1042-296X

IEEE-WG-802.11. (1999). IEEE Standard for Wireless LAN Medium Access Control and Physical Layer Specifications, *Technical Report*, IEEE WG 802.11, August-1999

IEEE-WG-802.11. (2002). Medium Access Control (MAC) Enhancements for Quality of Service (QoS), *Technical Report*, IEEE WG 802.11, November-2002

Kwon, D.; Woo, K.Y. & Cho, H.S. (1999). Haptic Control of the Master Hand Controller for a Microsurgical Telerobot System. *Proceedings of the IEEE International Conference on Robotics and Automation*, pp. 1722-1727, 0-7803-5180-0, Detroit, MI, USA, May-1999.

Lamping, U.; Sharpe, R. & Warnicke, E. (2006). Wireshark User's Guide. 2006

Live Networks. (2009). LIVE555 Library for Multimedia Streaming, *http://www.live555.com/liveMedia*

Luo, R.; Lee, W.Z.; Chou, J.H. & Leong, H.T. (1999). Tele-Control of Rapid Prototyping Machine Via Internet for Automated Tele-Manufacturing. *Proceedings of the IEEE International Conference on Robotics and Automation*, pp. 2203-2208, 0-7803-5180-0, Detroit, MI, USA, May-1999

MPEG-4. (2009). MPEG-4 library for coding/decoding, *http://sourceforge.net/projects/ffdshow*

RFC-3550. (2003). RTP: A Transport Protocol for Real-Time Applications. *IETF document*, July-2003

Sgouros, N.M. & Gerogiannakis, S. (2003). Robot Teleoperation Environments Featuring WAP-based Wireless Devices. *Journal of Network and Computer Applications*, Vol. 26, No. 3, August-2003, pp. 259-271, 1084-8045

VP7. (2009). VP7 proprietary video compression codec, *http://www.on2.com/technology/vp7*