Behind the Structure of Video VQ-Coder

Pavel HANZLÍK, Petr PÁTA

Dept. of Radio Electronics, Czech Technical University, Technická 2, 166 27 Praha, Czech Republic

Hanzlip@feld.cvut.cz, Pata@feld.cvut.cz

Abstract. *This paper introduces an implementation and structure of a video codec (coder-decoder) using Vector Quantization (VQ) in the program Matlab. It also aims on the VQ weak and strong features and it describes an experiment with turning the advantages into concrete profit in video data compression. We would like to aim also on usage in video – MPEG 4 (Motion Picture Experts Group) standard – or in compressing the scientific (high resolution) images. In conclusion we are comparing our approach with other image compression methods with objective and subjective criteria of image quality perception.*

Keywords

Vector Quantization, video processing, multidimensional space, image processing, uniform codebook reduction, codebook, compression, Matlab.

1. Introduction

Although nowadays we are easily capable to store large volumes of data, the idea of data compression is still up-to-date especially in data-type video - where we impact on need of processing large amount of data very fast on the incoming side of the communication chain. The redundance and irrelevance reduction was very well solved by the MPEG designers, but in today's multimedia society with demand for integration of large number of communication services into possibly most complex system we also search for ways to speed up the data transmission. One of possible solutions could be the usage of Multi-Dimensional Vector Quantization. The Vector Quantization (VQ) already found its use in sound and speech compression applications, in construction of psychoacoustic models and other specialized branches. Also some of its fundamental characteristics speak for its applying in video (see [6], [7]). Though probably the most important thing is to consider properly where to put the VQ into the signal compression process, how to optimize its use for particular transmission type and how to design a meaningful, complete conception of the coding chain to achieve the best results. Another important requirement is that the error between data incoming into the coder and outcoming from decoder is either minimal or not perceivable by the observer.

Vector quantization is a strong method for data compression when talking about textures and static images. The usage for example in scientific images (high bit resolution) or in video is still a good challenge. The MPEG 4 standard is an opened one, and the structure and design of the video codec including the VQ is an important part of it.

2. The Conception of Using the Vector Quantization in the "I" Frames Video-Sequence

Probably the easiest idea of the approach to the problematic can be done by using the VQ on the video stream compression formed by "I" frames. Some of the advantages from the static pictures can be applied on the video-sequence. The codebook is formed by specially designed set of sub-blocks of the particular frames of the sequence. This approach is researched in the following experiment, but there are also several other solutions. These could lead as a purpose for further experiments in the future.

Using of the vector quantization for example allows us to change the Discrete Cosine Transform (DCT) on the start of the standard MPEG coding chain for Karhunen-Loève Transform (KLT). The difficulties with using different KLT basis for each usage in the process can be replaced by a set of predefined vectors and coded by VQ.

Another example can be the usage of variable length (dimension) VQ applied immediately after the DCT calculation in the MPEG standard coding chain. The variable length vectors can here be formed from the zigzag reading of the non-zero coefficients in the 2D-DCT matrix.

We can also study the usage of Fractal VQ with variable dimension either in static images (see [2]) or in video *"I, B, P"* frames in the final MPEG stream. Another approach can be revised in the various ideas of using the VQ in time dimension. For example when using a buffer for several frames and form the vectors in time dimension. The vectors can either be formed from the values on particular coordinates *x, y* or by following the move prediction of various video objects (MPEG 4, 7).

When concentrating on the static images compression approach, we have to mention the iterative methods minimizing the compression error like Linde, Buzo and Gray realized - the LBG algorithm - or fuzzy clustering method (see Fig. 1, Fig. 2) which generate the codebook by several repeating steps (for more reference, see [1]).

If we know the statistic dissipation of the vectors in

the Euclidian space (*m*-dimensional vector) and the starting number of all vectors, we can form the LBG algorithm by repeatedly reassigning the reference vectors set according to the center of the particular Voronoi cell. The VQ is a relation Q from k-dimensional Euclidian space R^k into finite set Y from R^k. We have

$$Q : R^k \to Y \tag{1}$$

where

$$Y = \left(\hat{x}_i, i = 1,2,..., N\right) \tag{2}$$

is the set of predefined vectors and N is the number of the vectors in this set.

We define a distortion measure

$$D\left(x, q\left(x\right)\right) = \frac{1}{N}\sum_{i=0}^{N-1}\left(x_i - \hat{x}_i\right) \tag{3}$$

that represents the error of representing vector $\boldsymbol{x}$ by its model x'. This error can be iteratively minimized by changing the vectors in the so called „training" set.

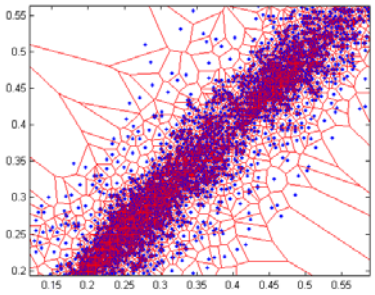

Fig. 1. The starting Voronoi diagram.

The centroid (part of the training set) is generated from n Voronoi cells, corresponding to the vectors dissipation. These two steps converge after several iterations to the minimum (square) error (MSE) criteria.

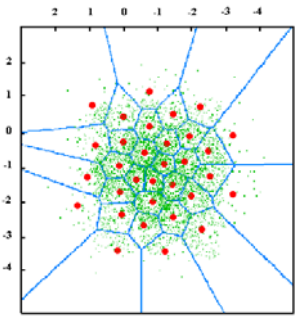

Fig. 2. Voronoi diagram with assigned centroids.

Each iteration though has two parts – assigning the codebook and adjusting it according to the image character. The adjusting is done iteratively by reassigning the particular training set with a new one. When achieving better results in MSE, the new one becomes the reference. This process converges and minimizes the quantization error**.**

2.1 *"K-means"* fuzzy clustering method

Here the idea is the same, but the approach in adjusting the codebook is simplified, the Voronoi diagrams needn't be calculated. The training set is predefined randomly, the iterativity criteria is formed by averaging the Euclidian distance in the *m*-dimensional space between the nearest neighbors in accordance with the weight of the present iteration count (Fig. 3, Fig. 4). This method also converges (earlier) to similar results.

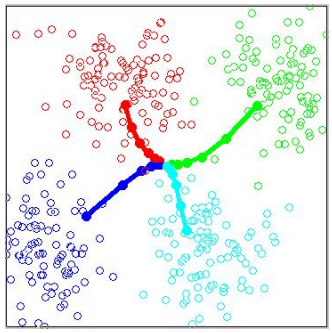

Fig. 3. Probability dissipation after the first 5 iterations of the *k-means fcm* method.

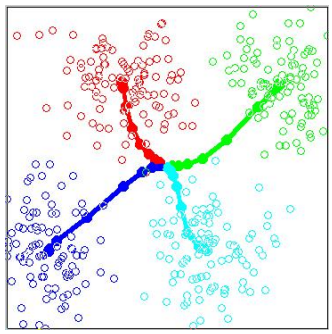

Fig. 4. After the last (8th) iteration of the *k-means fcm* method.

3. Experimental Part – Conception in Matlab

The overall concept (method) of the coder is a vector quantization with adaptable codebook, which is generated for the input sequence by alternative algorithm of uniform reduction of the codebook size. The input variables of the vector quantizer are vectors formed from sub-blocks of 4x3 pixels of the frames of the input video-sequence. Because we reason about color input video-sequence, the dimension of the quantized vector is 36 (three planes – *R, G, B*). In the experiment we are using first 100 frames of the video-sequence (25 fps – 8 s). We achieve here that the motives of the images are not changing much with time and so the codebook can be designed from all vectors of the sequence. Even if some theme is viewed only for a short period of time, it still is a part of the codebook. This approach can of course be modified by alternative image analysis, e.g. following the method until a frame cut.

The general structure of the experiment can be divided into several basic parts:

- Decomposition – The input color video-sequence is transformed to uniform dimensions (height, length) frames, which form the basic "I" frames of the coded sequence. After previous consideration the size was chosen comparable to some of professionally designed standards, nominally 435×343 pixels. The video-sequence is then formed by the particular frames with 25 fps frequency.
- Coder – The VQ algorithm input unit is a single vector of the original "I" frame. That is why the main function of a coder is the conversion of a single frame into vector representation of an image signal. The vector has dimension 36. The part of the coder in the experiment is also a controlling decoder that reverts the vector representation back into the image representation. The reverse algorithm verifies the lossless reversibility in this stage of compression
- Codebook generator – One of the most exacting parts of the program. All vectors of the 100 frames set are analyzed and by uniform codebook reduction we eliminate those vectors only, that won't be included in the final codebook used for compression. The quality of the image must not be decreased and the resulting codebook has to be optimized maximally to the allotted codebook bit size.
- Quantizer – The most exacting part of the program, when talking about time and number of calculations. For each vector of the codebook we allocate indicia and by a method of browsing the registry we are looking for a vector with minimum variance. There can be various criteria used, for example from usual mean square error and average error up to minimum sum error. According to the criteria used the best vector is chosen into the decoded image and its indicia are saved. The resulting indicia progression can be compressed by usual run length coding algorithms and finally prepared for data transfer.
- Decoder – The vector representation decoded from the indicia stream that is not that hard to decode by reverse algorithm used in the "coder" stage. The image and frame sequence can be viewed in this stage.
- Analysis – This part is linked to all of the previous stages and allows us to dynamically change the methods used in various coding stages to achieve the best results. The data obtained in quantizer stage are here also lossless coded either with RLE (Run Length Encoding) coder, or by Huffmann coding. The resulting bitrate and objective quality of the decoded images is calculated.

After previous research over the basic block size (1×2, 2×2, 8×8 etc.) the block formed by 4×3 pixels was chosen: four in length (horizontal) and three in width (vertical). Each pixel is with RGB coordinates represented by three numbers from the interval of integer numbers 0-255. That makes 8 bits for each coordinate, 24 in total. The basic block 4×3 pixels is here within represented by 36 numbers, sorted into vector.

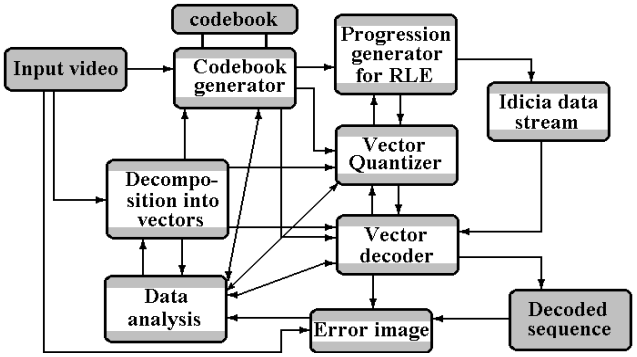

Fig. 5. Block structure of an adaptive VQ coder – decoder.

3.1 Uniform Reduction of the Codebook Size

The main problem in VQ is still the optimality of the used codebook. We design the codebook to achieve minimum mean square error between the coded and decoded image. The adaptive codebook VQ approximates the universal and optimal codebook character and is able to flexibly conform to momentary user's requirements. The sub-optimality in this case has to be invisible to the observer.

In the first phase of the codebook construction we collect all vectors from the predefined set of frames. This initial codebook is so called "complete". Here the highly correlated vectors form a representative basis of the "firstly reduced" codebook. The next reduction can be done by iterative optimizing algorithms that we usually use for VQ - LBG, GLA (Generalized Lloyd Algorithm) [1], [2], but these would highly lengthen the total compression time. These algorithms are easily able to achieve smaller compression error, but also more robust methods bring comparative results and in much shorter time.

We also have to reconsider that when viewing 25 fps sequence, with higher compression error, the human eye is still able to integrate the images and accept higher error rate than would be acceptable for a single static frame.

The "complete" codebook is reduced by elimination of multiple occurred vectors and we also choose only the most representative of the highly correlated. Here the compression error is very small and visually not recognizable after decoding. We still are able to reduce this "firstly reduced" codebook by choosing only the uniformly spaced vectors in the Euclidian space from the total set. Maximum codebook size is limited to user's quality requirements.

By previous empiric studies an approximating table was formed that shows the minimum "firstly reduced" codebook sizes to achieve minimum error by the uniform reduction and that results in most effective "final codebook" size (Tab.1).

The empirically detected "firstly reduced" codebook sizes are chosen by the following criteria: The vector not eliminated in the second phase is the best representative one from a concrete set of vectors (uniform cell).

Final codebook max. size	16 384	4 096	1 024	256	64	32	16
Firstly reduced min. codebook size	80 000	18 000	4 000	1 500	400	200	100

Tab. 1. The relation between the final and firstly reduced codebooks.

The concrete set of vectors does not spread further than mean square Euclidian space distance 150 [q^2] - (q is the quantization step of the scalar uniform quantization of RGB coordinates 0-255) from the representative vector. A violation of this criterion would cause visible and uncomfortable blocking character of single frames. We probably cannot eliminate this effect at lower codebooks sizes, but for the given codebook size we can distance this effect.

The last phase, mostly endangering the video quality, is the phase of elimination of the least distant vectors from the previous stage (by this we mean iterative elimination according to the local minimum square distance applied on the whole set of remaining vectors). The image can loose smooth changes over colors or details, but the image motive sustains. Image quality depends on user's requirements.

3.2 Adaptive Codebook Generating Algorithms

The adaptive codebook guarantees the sufficient decompression speed and also the minimum error between coded and decoded images. This has its origin in static images and textures compression. But as long we are not talking about static images but about a video stream, we have to reconsider this. The LBG and GLA algorithms are iterative. They have a heavy feedback to the original image. In the case of video, iteration would have to optimize all the video-sequence frames and that would lead to enormous extension of the coding time. The alternative and more robust algorithm of uniform codebook reduction allows us to choose effectively and fast the representative set of vectors to help to apply the VQ compression algorithm.

3.3 Indicia Coding (Data Part of VQ)

If the coding directive (codebook) is known to both coding and decoding side, we can code the video-sequence with help of pointer – indicia – on the correct place in the codebook area.

The indicia transfer can also be compressed by one of the lossless compression methods like RLE (Run Length Encoding), or Huffman entropic coding. Both the coder and the decoder need a buffer to preprocess the sequence or concrete transfer type.

The table of the experimental results shows good results especially in quality of the image reproduction and compression ratio in the range of codebook sizes from 32 to 1024 bits. E.g., for a very high quality of the decoded image (to 2 % average error) we're using a codebook of 512 elements, compressed with the ratio 1:38 (0.6 bpp). It is useful to compare VQ with the standard coding methods as can be seen in the following paragraph.

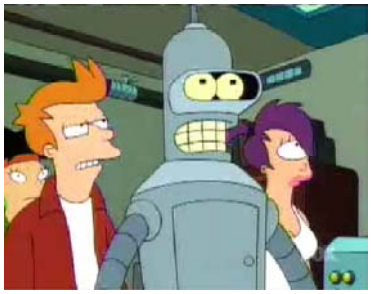

Fig. 6. The original (the first) image of a video sequence.

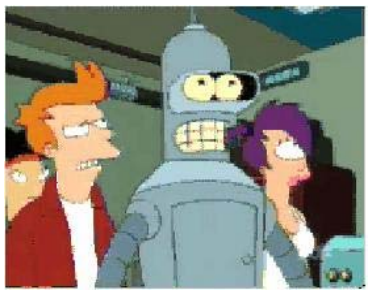

Fig. 7. The decoded image: codebook size 256 elements (2^8).

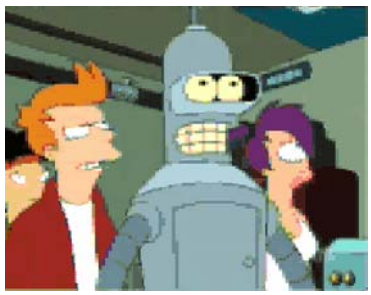

Fig. 8. The decoded image: codebook size 64 elements (2^6).

Fig. 9. The decoded image: codebook size 16 elements (2^4).

4. Comparison with Other Compression Methods

If we want to compare the used method with other static pictures coding methods, we can use Matlab with an advantage of creating a 3D simulation of the coder behavior in time. On *x*-axis of the following graph is the time

progress, on *y*-axis is the SNR (Signal-to-Noise Ratio), and *z*-axis shows different codebook sizes used.

Original video-sequence size	**41.628.195 [B]**
Bit rate of the original video-sequence	**11.190.375 [B/s]**

Codebook size [b]	2^x	Compressed video-seq. [kB]	Compression ratio		Reduced bit rate [kBps]	MSE [%]	Avg. Error [%]
16384	14	2063	1:20,1763	1.895bpp	555,68	6.47	1.06
8192	13	1873	1:22,2197	1.080bpp	504,49	7.33	1.22
4096	12	1770	1:23,5154	1.021bpp	476,24	7.83	1.32
2048	11	1575	1:26,4372	0.9078bpp	423,45	8.72	1.58
1024	10	1170	1:35,5683	0.6748bpp	314,68	9.80	1.78
512	9	1073	1:38,7885	0.6187bpp	288,56	10.13	2.07
256	8	888	1:46,8890	0.5118bpp	238,70	11.14	2.44
128	7	774	1:53,7537	0.4465bpp	208,25	11.83	2.74
64	6	565	1:73,7070	0.3256bpp	151,87	12.68	3.03
32	5	382	1:109,0312	0.2201bpp	102,66	13.02	3.24
16	4	279	1:149,1882	0.1609bpp	75,03	16.09	4.59
8	3	202	1:205,6601	0.1167bpp	54,43	20.69	6.98

Tab. 2. The complete measure table of experimental results.

If the codebook is well designed for the whole video-sequence, the SNR is approx. constant in the time dimension. As we can see from the data, although the SNR is not as high as in the static images compression methods, the compression is optimized to all frames in the video-sequence. In each plane of the RGB image with VQ we achieve SNR from 15 to 17 dB. Enlarging the codebook, the SNR rises.

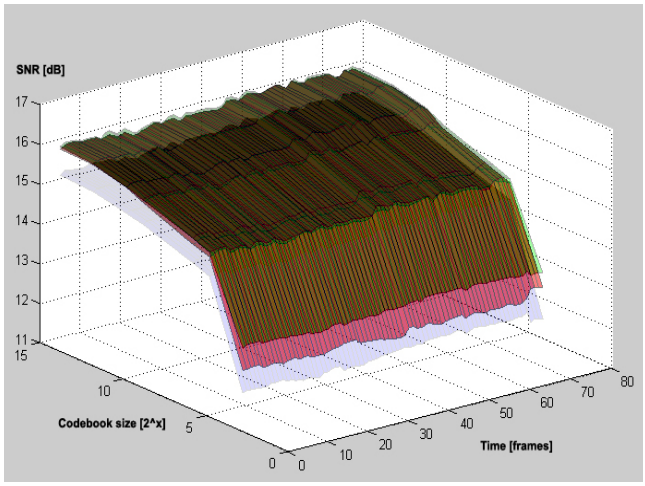

Fig. 10. R, G, B layers SNR in relation to time and codebook.

In comparison, a simple Fourier Transform (block size 16, bit depth 8) of the same input image, using 3 RGB layers individually coded, is reconstructed with SNR 33-36 dB. This measure is done only for the first frame of the video-sequence and does not optimize it, but it has significantly better results.

A Discrete Cosine Transform with the same characteristics (block size 16, bit depth 8) is worse (26 to 29 dB SNR) but still better than VQ applied directly on the image area.

Fig. 11. Walsh – Hadamard Transform SNR 26dB.

We've been studying also Wash-Hadamard Transform and Slant Transform, and found an interesting conclusion, that even if we achieve SNR 23 to 26 dB, resp. 24 to 25 dB with these transformations, the quality of the image seems to be worse to the observer (see attached photos with Walsh-Hadamard Transform SNR 26 dB and Slant Transform SNR 25 dB) than when using the VQ.

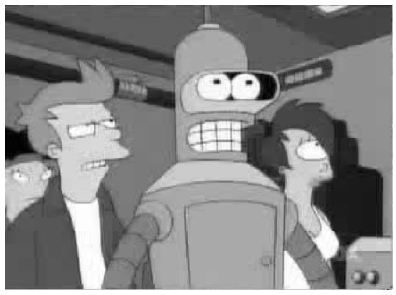

Fig. 12. Slant Transform SNR 25dB.

Fourier Transform seems to be the best method in this case; all others are comparable with the VQ on subjective quality criteria. Images coded with VQ look better to the observer, although the SNR is much lower. DCT, nor Walsh-Hadamard or Slant Transform cannot be compared with VQ when concerning on compression ratio. Achieving compression ratio from 2:1 to 6:1 with these methods, compression ratio of VQ in range from 25:1 to 40:1 is advantage.

Mentioned methods are integral transforms. This causes subjectively worse perception of image quality on block borders (N×N sub-blocks of image), object edges etc.

We have followed the usual standards for subjective quality assessment of ITU-T, "Methods for the Subjective Assessment of the Quality of Television Pictures", "Subjective Video Quality Assessment Methods for Multimedia Applications and "Video Coding for Low Bitrate Communication" (see [8], [9], [10]) when talking about the subjective quality measurements.

5. Conclusion

The adaptive codebook generating iterative algorithms for VQ are based on the idea of compressing static

images and textures. They have a strong feedback to the original image and very effectively reduce the quantization error. In case of data – type video – we need to reconsider the usage of VQ to achieve comparable results. Especially the structure of the coder is very important. There are still many approaches to MPEG 4 – an opened video standard – that could be useful when applying the advantageous VQ characteristics.

The compression times are still too high, but with the progress gradient in the information technology we will soon be able to use faster coding devices. The reduction of the transferred information can also bring a new view on UMTS technology and in usage of fast video applications.

Acknowledgements

This research and publication has been created at the Dept. of Radio Electronics, ČVUT – FEL in Prague (CTU) under the auspices of the research project "Kvalitativní aspekty kompresních metod obrazu v multimediálních systémech" and was supported by grant No. 102/02/0133 of the Grant Agency of the Czech Republic. The parts that deal with the design of compression methods for high speed video transfer were supported by the grant within the research intention of the psg. Project GAČR No. 102/03/H109 „Metody, struktury a komponenty elektronické bezdrátové komunikace".

References

[1] GERSHO, A., GRAY, R. M. *VQ and signal compression*. Kluwer Academic Publishers, 1992.

[2] ABUT, H. *Vector Quantization.* Piscataway: IEEE Press, 1990.

[3] KLÍMA, M., BERNAS, M., HOZMAN, J., DVOŘÁK, P. *Zpracování obrazové informace.* Praha: ČVUT, 1996 (in Czech).

[4] VÍT, V. et al. *Televizní technika.* Praha: SNTL, 1979 (in Czech).

[5] KOŠŤÁL, E. *Obrazová a televizní technika II. – Televize.* Praha: ČVUT, 1998 (in Czech).

[6] *Encyclopedia of Computing Science and Technology 29.* New York: M. Dekker, [1975- present] (supplement 14).

[7] *Encyclopedia of Computing Science and Technology 33.* New York: M. Dekker, [1975- present] (supplement 18).

[8] *ITU-T Methods for the Subjective Assessment of the Quality of Television Pictures.* ITU-T Recommendation 500-2, 1982.

[9] *ITU-T Subjective Video Quality Assessment Methods for Multimedia Applications.* ITU-T Recommendation P 910, August1996.

[10] *ITU-T Video Coding for Low Bitrate Communication.* ITU-T Recommendation H.263; version 1, Nov. 1995; version 2, Jan. 1998.

About Authors...

Pavel HANZLÍK was born in 1979 (Děčín, Czech Rep.), in 2003 he graduated at CTU, Prague. He continues in the studies on the PhD. program and his main interests are image and video compression, theory and comparison of used methods, and practical implementation of the new ones developed in his research work.

Petr PÁTA, PhD. was born in 1973 (Prague, Czech Rep.). In 1997 he graduated at the Mathematics-Physics Faculty of the Charles University in Prague and obtained an academic title Mgr. (M.Sc.) for physics. In 2002 he achieved title Ph.D. in Radio electronics at CTU-FEL in Prague. At present he works as a research professor assistant at the Dept. of Radio Electronics, CTU-FEL, Prague. His research and interest aim on photonics and scientific data compression and processing.

RADIOENGINEERING REVIEWERS

September 2004, Volume 13, Number 3

- BALÁŽ, I., Slovak Univ. of Technology, Bratislava
- BIOLEK, D., Military Academy, Brno
- ČERNOCKÝ, J., Brno Univ. of Technology, Brno
- ČERNOHORSKÝ, D., Brno Univ. of Technol., Brno
- DÝMAL, P., Brno University of Technology, Brno
- HANUS, S., Brno University of Technology, Brno
- KASAL, M., Brno University of Technology, Brno
- KOVÁŘ, P., Czech Technical University, Prague
- KOZUMPLÍK, J., Brno Univ. of Technology, Brno
- KRATOCHVÍL, T., Brno Univ. of Technology, Brno
- KRŠEK, P., Brno University of Technology, Brno
- KURTY, J., Military Academy, Liptovský Mikuláš
- KVIČERA, V., TESTCOM, Prague
- LAIPERT, M., Czech Technical University, Prague
- LEONE, M., Siemens Corporate Technol., Erlangen
- LEVICKÝ, D., Technical University, Košice
- MÜLLER, L., University of West Bohemia, Pilsen
- RAIDA, Z., Brno University of Technology, Brno
- SOVKA, P., Czech Technical University, Prague
- SÝKORA, J., Czech Technical University, Prague
- ŠEBESTA, V., Brno Univ. of Technology, Brno
- ŠKVOR, Z., Czech Technical University, Prague
- WIESER, V., University of Žilina, Žilina
- YMERI, H., Katholieke Universiteit, Leuven