Animated Image Compression in CATSERVER - Preliminary Description

Georges Gyory

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

Animated Image Compression in CATSERVER - preliminary description

Georges Gyory

N° 3457

Juillet 1998

PROGRAMME 1

ISSN 0249-6399

Animated image compression in CATSERVER - preliminary description

Georges Gyory*

Programme 1 : Networks and Systems

Projet HIPERCOM

Rapport de recherche n˚3457 - Juillet 1998

12 pages

Abstract:

A scalable moving image compression system is proposed for the CAT-SERVER application, MPEG-like in its Group Of Frames structure for predictive coding but using supplementary information retained from previous images to help image prediction and a novel movement detection / coding scheme based on spanning points with associated motion vectors.

Key-words: animated image compression, MPEG, motion detection, motion coding, predictive coding, background image

(Résumé : tsvp)

* Email: Georges.Gyory@inria.fr

Unité de recherche INRIA Rocquencourt,
domaine de Voluceau, Rocquencourt, BP 105, LE CHESNAY Cedex (France)
Telephone : (33 / 0) 1 39 63 55 11- Fax : (33 / 0) 1 39 63 53 30

Compression d'images animées dans CATSERVER - description préliminaire

Résumé : Cet article décrit le système de compression d'images animées proposé pour CATSERVER. Il réalise un compromis qualité d'image - taux de compression réglable utilisant des frames prédictives comme en MPEG, mais se sert d'informations supplémentaires gardées des images précédentes pour améliorer le taux de compression et utilise une nouvelle méthode pour détecter et coder le mouvement, basée sur un ensemble de pairs point de l'image - vecteur de mouvement.

Mots-clé : compression d'images animées, MPEG, détection de mouvement, codage de mouvement, codage prédictive, arrière-plan.

1.0 Requirements:

The CATSERVER cable-TV system needs an animated image and sound compression scheme that is

1. asymmetrical (ie. fast to decode while encoding time matters relatively few)
2. achieves MPEG-like compression rate on slow motion images
3. is scalable with respect to the client's CPU capacity
4. is scalabe with respect to the available transmission bandwidth (with several differently compressed versions being prepared and stored on the server side)
5. can switch between the said versions during transmission to adapt to a lower/higher bandwidth available.

In this paper I shall discuss the proposed image compression scheme.

2.0 Image compression

The bandwidth of a TV emission (168 and 199 Mbits/sec uncompressed in NTSC and PAL respectively, over 1300 in HDTV - see (3)) makes it necessary to use compression for transmitting it on a network. Standard compression methods applied on individual images giving unsatisfactory results, compression must exploit the similarity between consecutive images. What is left to transmit is then compressed usually by entropy coding - JPEG in MPEG, but for CATSERVER wavelet compression (4), pattern matching compression (1) and fractal compression (2) are being considered too.

2.1 Groups of pictures

Consecutive images with similarities are coded (compressed) as groups of pictures (GOP) into groups of frames (see definition later). The structure of a group of frames is similar to that of MPEG.

Predictive coding uses motion detection both in MPEG and in the CATSERVER system (but predictive coding can be done without motion detection). Motion detection means that in order to better compress the following image, some regions of (previous) reference images are copied in the next

image and shifted by a motion vector instead of reproducing them from scratch.

Images are sent in three types of frames:

1. Intra-frames (I-frames) are self-contained. The first image must be sent in an I-frame as there is no reference image to use on the receiver side yet. Another role of the regularly sent I-frames is to stop error propagation. A reconstructed image on the receiver side may contain errors that arise either due to a transmission error or due to floating-point arithmetic executed on different processors with different exactitude. These errors will not be corrected in the following P- or B-frames as they are not present on the sender side. An I-frame sent will put things back in order.

2. Predicted (P-) frames rely on the last I- or P- frame sent. The following image is coded with respect to the last image reconstructed on the sender side (and will correct its errors). Regions of the last (already tranmitted) image are shifted according to motion vectors and then a pointwise difference is calculated between this (predicted) image and the following one (that to be transmitted). The description of the motion regions and the motion vectors are transmitted with the compressed pointwise difference image. As the pointwise difference can be compressed more than the whole image, we gain on the compression ratio.. Transmitting the pointwise difference between the new image and a reconstructed reference image is called **predictive coding**.

3. Bidirectionally predicted (B-) frames are coded similarly, but using both the preceding and the next I- or P- frame. This enhances the compression rate even more, but not for very long B-frame sequences. For this reason and in order to avoid more error propagation, usually there are no more than two of them in a row. B- frames are never used in the prediction of other frames.

Image sequences are coded in a sequence of frames like

I B B P B B P B B P B B I B B P B B P...

but the frames are sent in a different order to facilitate decoding. As the following I- or P- frame is needed to decode a B- frame, the order of emission is like

I P B B P B B P B B I B B P B B P B B ...

, with P-frames (and all but the first I- frame) hopping before the B-frames that they follow in the sequence of their display. This type of coding needs more memory and introduces a delay in decompression.

In the case of a cut in the sequence (an image totally different from the previous one) a new I- frame is sent. Pictures corresponding to frames from an I-frame to (but not including) the next I- frame are called a **GOP**.

More detailed description of the GOP structure can be found in most descriptions of MPEG like (5) and (3).

2.2 The motion detection problem

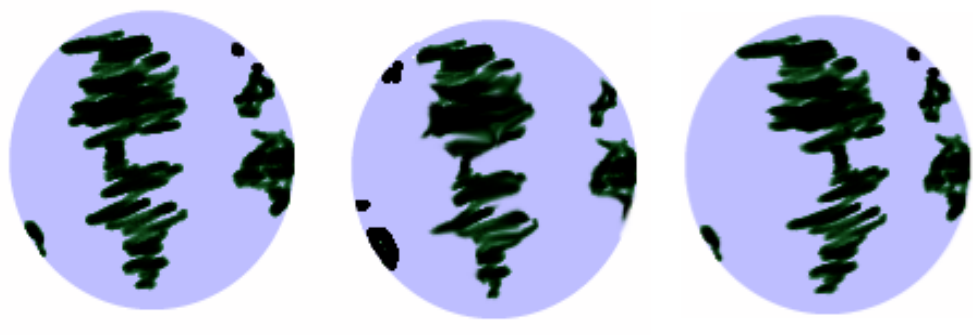
The next problem is, how to detect and how to code motion if we are to use it. More exactly, how to detect

- regions that are displaced from one image to the next
- but only **approximatively** identical to the displaced region (identical from the practical point of view of the gain in compression rate)
- and their motion vector.

This approach leaves us with two problems. The first, that of the same region content multiplying in the next image, is resolved by the use of **backward motion vectors** (attached to regions in the next image and pointing to regions in the previous one, these possibly the same). The second problem is that the motion vectors are best found if you know the regions (recall that the identity of contents is approximative) but the regions are best found if you know the motion vectors already.

Methods used for motion detection are often either crude or computationally demanding. Some of them simplify using just a quadtree division of the image to find the regions they will try to match. Others use linear regions (in the sense the image is scanned) and look for pointwise correspondence (with respect to a possible constant shift and a quadratical error) (1).

It is possible to (often) simplify (for practical purposes) motion detection by recycling the motion vectors used to encode the last image, but the problem is still unsufficiently resolved, especially for the first image change.



Figs 1a, 1b, 1c

Consider now *Figures 1a, 1b* and *1c* supposed to represent a globe in free-hand drawing. The moving regions and the movement vectors are the same in the subsequent image changes (see *Figure 2*).

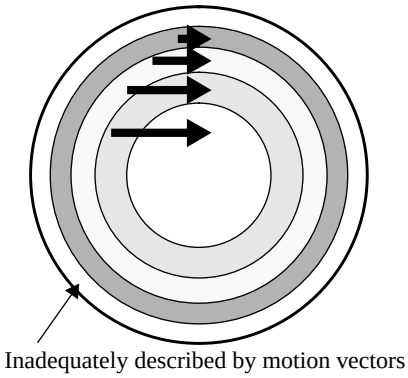
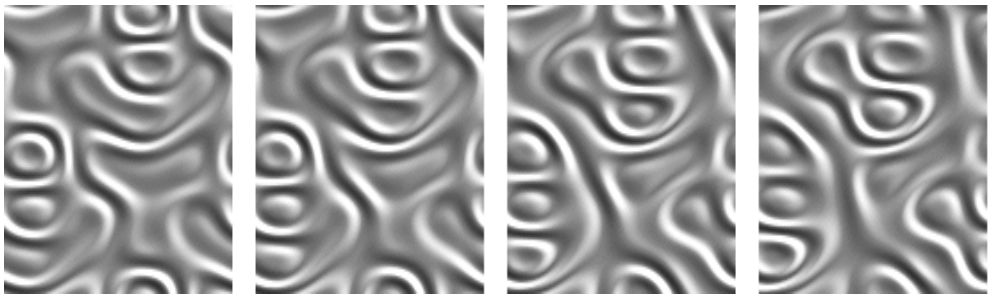


Fig. 2: Turning sphere: regions of motion with their motion vectors

First notice that motion vectors do not resolve all our problems. In the (white) peripherical region probably no motion will be detected at all. The left half of the image (supposing the globe is turning to the right) will become more detailed, the image approximated coarsely by motion. In the external part (in white in *Figure 2*) new details will appear and disappear, like New Guinea and Kamtchatka on the left and Ireland on the right of *Figure 1b.* Last but not least, changes in the motion vector are continuous in the circle (unlike in *Figure 2*) and may amount to fragments of pixels. Depending on the texture, motion of non-integer pixel values may go undetected. On the other hand, motion may be detected where there is no corresponding motion on the turning sphere, ex. Kamtchatka approximated by something

quite different (though in practice you don't look for very long motion vectors).

Figure 1c illustrates the quality of image obtained (but not the exact motion) if motion is coded by a continuous motion vector field (supposing no problems in detection and lots of regions coded) - courtesy of the "Dispersion" Photoshop filter.



Figs 3a, 3b, 3c, 3d

Figures 3a, 3b, 3c, 3d come from a test sequence 25 images long.* Notice that motion regions, though changing in a continuous manner, change their shape, are divided and united among themselves during subsequent image transitions.

2.3 Proposed solution for motion detection

The easiest way to start motion detection is to have previous motion regions and motion vectors. In lack of these, we must choose a salient rectangular part in the image and hope we can find it in the next image. The size of this part depends on the abundance of detail in the image (estimated ex. by the number of local maxima and minima on a scanning line) and in the case of *Figure 3a* may be about one third by one third of the image and it should contain contrasted parts with vertical and horizontal edges. Consider ex. the left side of *Figure 3a* about at the half of its height.

Prepare now six histograms by summing row- and column-wise the Red, Green and Blue values in the selected part, then six similar ones using the whole image. Doing a chi square test between histograms of the same kind

* The sequence was generated using the BackgroundWarper shareware program for Macintosh PPC - see ex. at http://www.shareware.com

shifted one along the other, the position of the best match between the two gives a guess on the vertival or horizontal component of a corresponding motion vector. This can be confirmed or infirmed by chi square test results for other components in the same direction. Hopefully the vertival and horizontal componenets will give a motion vector which corresponds to some real motion in the image. (Artificial counterexamples are always possible, but the chi square tests usually give several peaks and one of their combinations should work out well.)

The same histograms can be used to detect these salient regions at points where variations are sharp. In natural images if two or three promising places are found on each axis, chances are that at least on of intersections are of interest for motion detection. Notice that this may be untrue on artificial images: in each of the *Figures 3a, 3b, 3c, 3d* (the original images are in colour, with yellow and light voilet light sources on greenish paper) if histograms are made summing diagonally on the length of a diagonal (imagine the contents of one of these figures scaled to be square, a bigger image paved with it, then turned by pi/4, than a tile the size of the diagonal of the scaled tile cut out of it before building histograms as before), all the sums will be identical as the heights of two points with half the diagonal in between always cancel out.

Now that we found an approximate (rectangular) region with a displacement value we shall refine this region next. Starting with a matching point as the refined region (with respect to the displacement value and an error metrics), we examine its neighbours. In order to be added to the region, they should fit either with the exact displacement or with an error of 1 or 2 pixels in the displacement and an even better fit by the error metrics (the metrics with respect to displacement has yet to be calibrated). We continue exploiting the displaced region iteratively, like filling it with a paint pot in a paint program.

Once the region is established, we explore new regions starting with points on its border which had a displacement error, but with the modified displacement this time. This will expand the region we can track from one image to another. We continue exploring recursively. At the end, we have a moving region isolated from its background and from other regions unless we hit into the background, in which case most of the separation work is done. The number of times we may pass over a point is bounded if we allow no increase in approximation errors.

Then we delete the explored region from both images and build new histograms to identify other salient parts and moving regions until we find no more. In the end of the process we usually have one background, some moving regions and some regions we couldn't code by motion.

The process is relatively simple and permits the use of relatively complex metrics which allow for errors in the motion vector. It uses time *(coarse estimation in parenthesis)* essentially for

1. buiding histograms *O(number of pixels * (regions identified + 1))*
2. matching histograms *O(less than the linear size of the image)* and
3. expanding regions *O(the number of pixels).*

2.4 Motion coding

Once we have motion (defined by regions and motion vectors), the question is how to code it. Supposing we have a description by a grid of points in the image with motion vectors associated to them, we can reconstruct motion at least in two ways: either we give the motion vector of the grid point to every point in its part of the Vornoi diagram of the grid or we interpolate linearly inside every triangle of the Delaunay triangulation of the grid (let alone techniques of fuzzy control).

One useful algorithm is described in (2) (for fractal compression):

1. We start by defining a uniform triangular grid of points over the image and associate to each point the motion vector of its region (with a BIG value for non-motion-coded parts).
2. We evaluate locally the accuracy of this description of motion by a metrics.
3. In the areas where approximation is not good we add points to the middle of each arc.
4. Then we try to suppress points (either previous or some of the newly added) and re-evaluate before confirming. If still unsatisfied after the suppressions, go to 3. Else end of algorithm.

On stopping we have a set of points with associated motion vectors that describe motion in the image transition. This encoding is scalable by

employing a more or less permissive metric (with respect to pointwise or displacement errors).

2.5 Background layer

We shall use the extra memory available (with respect to a standard MPEG decoding card) to keep in memory the background layer of the image to avoid complete retransmission of its recently uncovered parts which had been seen before.

The background layer of an image has, heuristically, two properties:

- it is covered and uncovered by moving regions
- its area is big (with respect to the moving regions).

These properties (as all the motion detection business) are strictly independent of the image contents (real-world objects, focus or semantics of any kind) and permit the identification of the background by the following rules:

1. on sending/receiving a big image (i.e. after a cut in the movie) the background is reset to middle gray

2. after the first image change, one of the movement regions (ex. the biggest in horizontal/vertical size, moving slowly, surrounding other regions) will be chosen as background and its contents copied into the background buffer

3. after the subsequent image changes, the contents of the same (background) region are copied into the background buffer. Notice that unupdated parts of the background buffer may contain data from the previours image changes.

On coding/decoding the next image, we use the background buffer in the prediction phase. Its contents are copied in the predicted image first, then motion regions added and the resulting image used as predicted image (for pointwise comparisons). Note that the contents of the background layer will be used only in the “holes” left in the previous image by motion.

The adavantege of this procedure is that parts of the background that were seen in an earlier image, then covered by a moving region and then uncovered will be recovered in the predicted image, avoiding their retransmission. This may be quite useful in the case of an object traversing a static back-

ground: when we shall have seen all of the background, all the image change can be described by the motion coding.

Of course, this procedure is possibly erroneous. The wrong region can be taken for the background. When the background region breaks up (a hitherto immobile object starts moving), further choices must be made. Our claim is that transmission rate can but be enhanced (and cannot be deteriorated) by it, at the only cost of more computing. In the case of a moving background (ex. the sea with waves), the gain can be important too as even if the shape of the waves isn't reconstructed correctly, the colour of the sea will be.

Detecting and keeping in memory more than one layers of the image seems to be more costly, less rewarding and very error-prone, therefore it will not be discussed here.

In B-images bidirectional prediction can be used (like in motion detection), using the direction which gives better results.

3.0 Conclusion

The methods described in this paper haven't been verified experimentally. This work is yet to be done and will be described in further publications. For the moment, we have reason to believe that

- there is ample space to increase motion detection and coding and the methods above provide a useful alternative to the usual methods for doing it
- more memory and some increased CPU power on the receiving side can be put to good use to increase the compression rate over MPEG in the case of slow-motion natural images (most of the programs to be transmitted).

4.0 References

1. *M. Atallah, Y. Genin, W. Szpankowski:* Pattern Matching Image Compression *Proc Intl Conf on Image Processing, vol. 2 pp. 349-352 (Lausanne 1996)*

2. *F. Davoine et al.:* How to improve pixel-based fractal image coding with adaptive partitions *in Jacques Levy Vehel et al. : Fractals in Engineering, Springer 1997*

3. *B. G. Haskell et al.* Digital video: an introduction to MPEG-2 *Chapman & Hall, 1997*

4. *S. G. Mallat:* A Theory for MultiresoutionSignal Decompositin: The Wavelet Representation *in IEEE Trans. on Pattern Anal. and Machine Intell. vol. 11 No. 7 July 1989, pp. 674 - 693*

5. *J. L. Mitchell et al.* MPEG Video Compression Standard *Chapman & Hall, 1996*

Unité de recherche INRIA Lorraine, technopôle de Nancy-Brabois, 615 rue du jardin botanique, BP 101, 54600 VILLERS-LÈS-NANCY
Unité de recherche INRIA Rennes, IRISA, Campus universitaire de Beaulieu, 35042 RENNES Cedex
Unité de recherche INRIA Rhône-Alpes, 1655, avenue de l'Europe, 38330 MONTBONNOT ST MARTIN
Unité de recherche INRIA Rocquencourt, domaine de Voluceau, Rocquencourt, BP 105, LE CHESNAY Cedex
Unité de recherche INRIA Sophia-Antipolis, 2004 route des Lucioles, BP 93, 06902 SOPHIA-ANTIPOLIS Cedex

Éditeur

Inria, Domaine de Voluceau, Rocquencourt, BP 105 LE CHESNAY Cedex (France)

ISSN 0249-6399