MERGING MICROSCOPY COMPRESSION TECHNIQUES WITH CONTINUOUS IMAGE REPRESENTATION

PAUL COCKSHOTT

Over the last 3 years Paul Cockshott, Yegang Tao and Gang Gao have developed a 3d pyramid based compression technique. This technique has been particularly targeted at microscopy image compression but is applicable to other medical scanning technologies where the raw data set can be viewed as a 3dimensional image. The original system was developed under a Proof of Concept award and resulted in the package MicroStack. This was reported on in [1].

The basic technique has the following advantages:

(1) It produces images that are free from ringing and blocking artefacts that characterise many other compression techniques.
(2) It achieves very good rate/distortion characteristics when compared to other compression techniques for microscopy data. That is to say, at a given bit-rate the signal to noise ratio will be better than with MPEG or JPEG.
(3) It is a true 3D technique, so that it allows the compressed image stack to be reconstructed along any of the axes x,y,z. It even allows one to view the data at an arbitray x,y,z angle.
(4) When compared to MPEG the distortion level is more even accross frames. With MPEG quality is markedly with keyframes than intermediate frames.

The technique has subsequently been improved by Tao as part of his PhD work, and these improvements are described in [2, 3]. These improvements relate to the use of more sophisticated vector quantization techniques and to the implementation of a distortion constrained rather than rate constrained compressor. In other words it allows one to specify the signal to noise ratio you want to achieve and then allocates the number of bits to the representation required to achive this.

1. APPLICABILITY

Let us consider to what extent it meets the needs of IP-Racine WP6.

Pro.

(1) The system is well proven and has be reported in the peer review literature.
(2) It can be readily applied to 3d data from other sources such as film or video, and we have in fact tested it on such data.
(3) It would give us a compressed representation allowing longish film sequences to be processed in memory.
(4) Because it uses a bandpass filtered pyramid structure, the system is free from blocking artefacts.

Con.

(1) The system as it stands is optimised for microscopy compression. It has been demonstrated as video compressor but in this context the bit rate is somewhat

higher than MPEG. These tests were done before Taos improvements to the algorithm and it may now be somewhat better for video than it originally was.

(2) It is not scale independent since the underlying technology relies upon the use of a codebook of pixel based patches at the lowest level. This means that expanding images to pictures of much larger scales would at present have to be done using pixel interpolation.

I consider however that it should be possible by a two step process to adapt this approach to a working cine representation. The first step would be to address the scale independence, since this is a specified design objective of the IP Racine work. As a second step one could introduce some form of motion compensation which would raise the compression ratio to one comparable to or better than MPEG. High level of compression is not an explicit design goal, but it would be an undoubted plus fpr the representation.

In order to explain how we can incorporate Sylvan's scale independent techniques into this framework, I will give an outline of the algorithm in its simplest form.

I suggest that readers consult reference [1], which is available on the departments bibliography server, for background information.

2. TYPES USED IN THE OUTLINE ALGORITHM

```
const
    red =1;
    green =2;
    blue =3;
    celltop = 4;
    depth =4;
type
    colour = red ..blue ;
    cellstep = 1..celltop ;
    cell =
    array [colour ,cellstep ,cellstep ,cellstep ] of pixel ;
```

A cell is a time space block of pixels that is the basis on which compression is performed

```
    block(x,y,z:integer ) = array [1..x ,1..y ,1..z ] of cell ;
```

A block is a larger time/space unit of information that is used in compression.

```
    blockpntr = ^ block ;
    blockpyramid = array [1..depth ] of blockpntr ;
```

The following defines the format of the compressed data

```
    codeindex = integer ;
    codeblock(x,y,z:integer )= array [1..x ,1..y ,1..z ] of codeindex ;
    codeblockpntr = ^ codeblock ;
    codepyramid =array [1..depth ] of codeblockpntr ;
```

```
    pyramidpair = record
        raw : blockpyramid;
        cooked : codepyramid;
    end ;
```

3. FUNCTION INDEX

Here are the headers of the functions we will use:

```
var
    codebook: array [0..511] of cell ;
function encode ( var c :cell ):codeindex ; (see Section 4 )
function decode ( i :codeindex ):cell ; (see Section 5 )
function shrink ( var b :block ):blockpntr ; (see Section 6 )
function expand ( var src :block ):blockpntr ; (see Section 7 )
function decodelevel ( l :integer ;var p :pyramidpair ):blockpntr ; (see Section 8 )
procedure encodelevel ( l :integer ;var p :pyramidpair ); (see Section 9 )
```

4. ENCODE

```
function encode ( var c:cell ):codeindex ;
begin
```

Ignore how this is done for now, assume it is some vector quantisation algorithm. Dummy result now.

Questions to address here are:

(1) Do we encode r, g and b components as a single parameterised function or vector.
(2) Do we encode them as three distinct functions - which is less efficient from and information theory point of view but probably faster to implement.
(3) Should energy thresholding be applied to map low energy input vectors to the null vector, which makes the resulting representation more efficient to serialise.

```
    encode← 0;
end ;
```

5. DECODE

```
function decode ( i :codeindex ):cell ;
begin
```

Again just assume this is some form of lookup algorithm

```
    decode← codebook_i;
end ;
```

6. SHRINK

```
function shrink ( var b :block ):blockpntr ;
begin
```

This takes a block and averages and subsamples it to give a smaller block, which would typically be half the size. For the moment this is a dummy result. Tao has done a lot of work on the best filters to use to minimise band crosstalk in this process.

```
    shrink← nil;
end ;
```

7. EXPAND

```
function expand ( var src :block ):blockpntr ;
begin
```

Take a block and expand with pixel interpolation to double the size. When we have the continuous representation, we would use that to handle the interpolation.

```
    expand← nil;
end ;
```

8. DECODELEVEL

```
function decodelevel ( l :integer ;var p :pyramidpair ):blockpntr ;
var
    Let b ∈ blockpntr;
begin
```

The top of the tree is uncompressed.

```
    with p do
        if l = depth then decodelevel← raw_l
        else
        begin
            (* Create a buffer to hold the result. *)
            new ( b ,raw [l ]^ .x ,raw [l ]^ .y ,raw [l ]^ .z );
            b↑← decode (cooked_l↑);
            decodelevel← b;
        end
end ;
```

9. ENCODELEVEL

```
procedure encodelevel ( l :integer ;var p :pyramidpair );
var
```

```
    Let b, b1, b2 ∈ blockpntr;
    Let c ∈ cell;
    Let x, y, z, i, j, k ∈ integer;
begin
    if l < depth then
        with p do
        begin
            b← raw_l;
            x← b↑.x;
            y← b↑.y;
            z← b↑.z;
            raw_{l+1}← shrink (raw_l↑);
            encodelevel (l + 1, p);
            b1← decodelevel (l + 1, p);
```

We now form a difference image between the decoded expanded level above and the current level.

This is an absolutely critical feature of the algorithm. This is what makes it a closed loop feedback system. Errors generated in the level n+1 of the pyramid are corrected by what is done at level n.

```
            b2← expand (b1↑);
            b2↑← b↑ - b2↑ ;
```

Create space for the cooked result

```
            new ( cooked [l ],x ,y ,z );
```

now initialise the cooked array by mapping b2 through the cell encode function

```
            for i← 1 to x do
                for j← 1 to y do
                    for k← 1 to z do
                    begin
                        c← b↑[i, j, k];
                        cooked_l↑[i, j, k]← encode (c);
                    end ;
        end
    end ;
```

This is the very simplest version of the implementation with no description as yet of how the vector quantisation, shrinking or exanding takes place.

10. DEVELOPMENT

I envisage development following a 3 stage process as shown in the accompanying diagram 10.1.

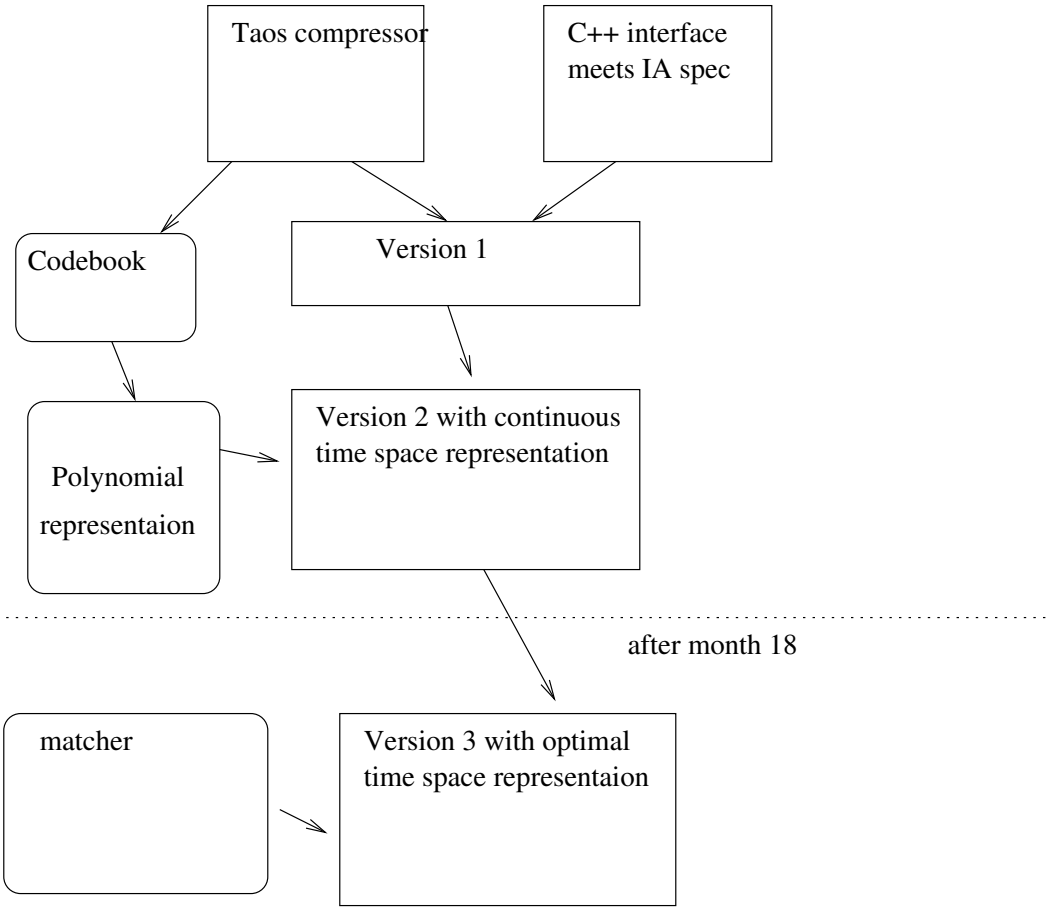

FIGURE 10.1. Workplan

We first need to have an abstract datatype for the image class that hides the implementation and allows it to interface with existing packages.

We then need to hide Tao's implementation behind this abstract datatype so that it can be tested with the framework of whatever user interface we chose. That is version 1.

Version 2 will involve replacing Taos vq codebooks with codebooks using some form of 3d polynomial or other mathematical function that allows continuous representaion in time and space.

Version 3 will incorporate the matcher to allow an enhanced design of pyramid that incorporates motion compensation and information from the higher levels of the pyramid. This should end up with a file format that is comparable in compression terms with existing state of the art techniques but which allows resolution independence.

Things to which particular attention may have to be paid in the final version are the ultimate file format. If we intend to use memory mapping, we will have to go through our datastructures and make sure that the ones that will be file mapped into memory do not contain any store addresses, but instead contain pointers in the form of offsets from the start of the file.

REFERENCES

[1] Confocal Microscopic Image Sequence Compression Using Vector Quantization and 3D Pyramids, Cockshott, W. P., Tao, Y., Gao, G. Balch, P., Briones, A. M., Daly, C., Scanning - The Journal of Scanning Microscopies, pp 247-256, 2003.

[2] Microscopic Volumetric Image Data Compression Using Vector Quantization and 3D Pyramids, Cockshott, W.P. Tao, Y. Gao, G. Daly, C. ,Picture Coding Symposium 2003 (PCS03), Saint Malo, France, pp 119-124, IEEE Computer Society Press.

[3] 3D Microscopic image coding by finite-state vector quantization in an enhanced image pyramid, Tao, Y. Cockshott, W.P. , In Proceedings of SPIE Conf. on Medical Imaging (2004), San Diego, USA .