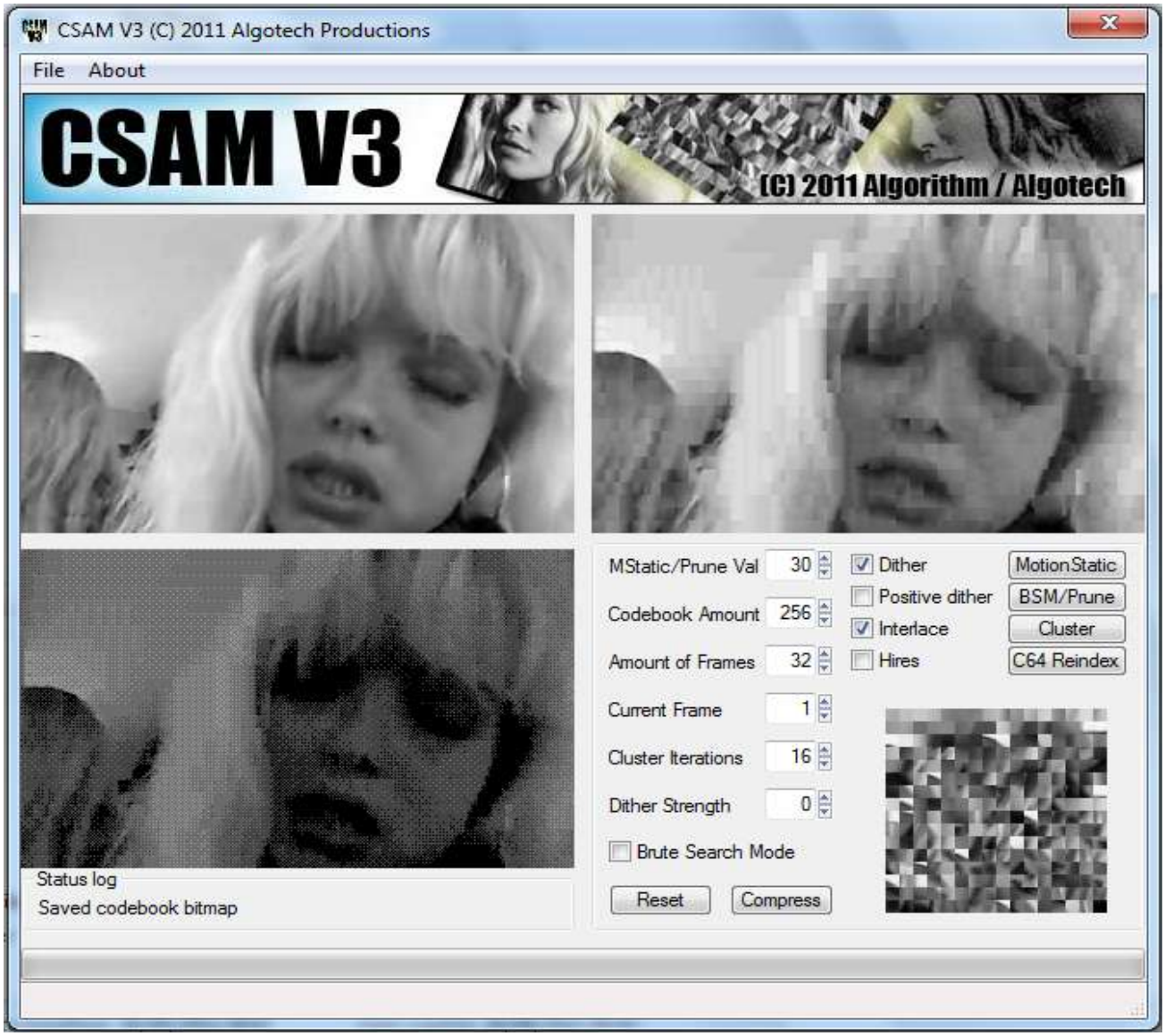
CSAM (Char Selection And Match) V3 - (C) 2011 Algorithm / Algotech Productions

INTRODUCTION

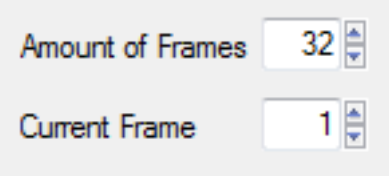
This Windows application allows the user to import any image/video and to convert and compress it to a format that can be played back on a Commodore 64.

The application utilises a method of compression known as Vector Quantisation. This method of video compression was introduced on the PC quite some time ago with one of the codec's being known as Cinepak. Main advantage of this compression method is that decompression is extremely fast! An Ideal candidate for the Commodore C64.

THE APPLICATION



Above are the main options that are available in the application. Extreme attention needs to be taken on two fields which are the below

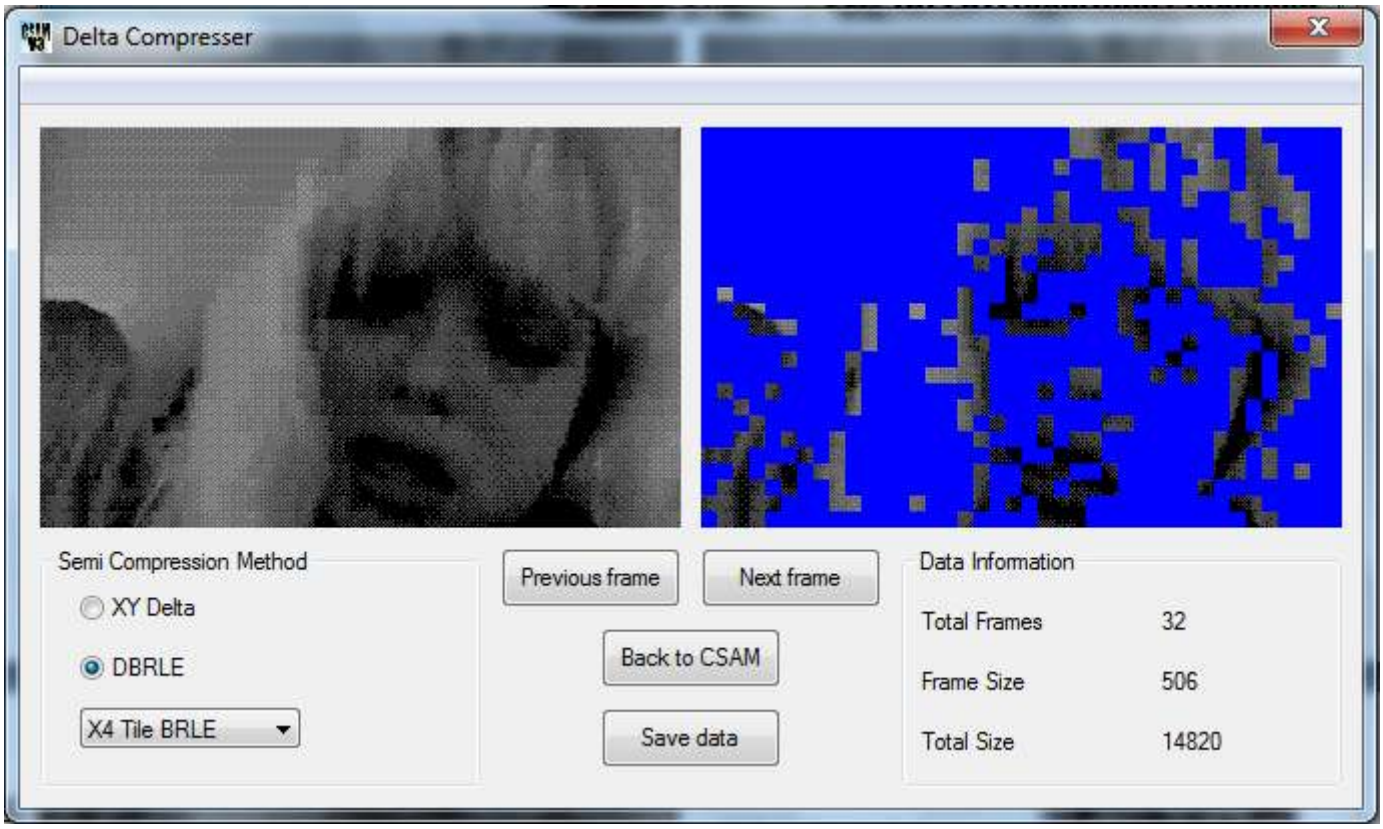


All loading/conversion / encoding functions will only operate on the data entered above. This allows great flexibility as the user can import for example 30 frames from frame 8 to 37 and then import another video from frame 40 to 50 etc.

BASIC TUTORIAL (MAIN APP SECTION)

1. Specify amount of frames to load using the '**amount of frames**' section. In this example '**amount of frames** is at 32
2. Select the '**load avi'** option in the menu bar and open the avi file. This will now load 32 frames from frame 1 to 31
3. Select '**Mstatic/Prune'** value of 8 and click on the '**motion static'** button. This will compare future frames and keep 'slightly moving' 8x8 blocks static across frames. The higher the number, the more static each frame will be. This will increase compressibility in the process. Experiment with these figures by selecting and then moving through frames to determine whether it is the optimal setting. If the results don't look too well, you can load the avi file again or select the '**reset**' button which will reload the original data from the memory bank. Concentrate more on the C64 preview image as the output.
4. There are two codebook generation methods in this application. By default, the Prune mode is selected However when ticking the '**brute search mode'** this will then revert to the exhaustive method. In this example, the brute search mode is unticked.
5. Click on **BSM/Prune**. The routine will then look through each 8x8 block in the image and frames utilising the value in **Mstatic/Prune** value. Using zero as the value will place the first unique 256 codebooks even if they are very similar to each other. The point here is to increase the value until every frame is utilised. If the codebook is entirely filled up, increase the value and click on **BSM/Prune**. Repeat this process until there is indication that all frames have been utilised with one/two gaps at the end of the codebook.
6. Now that the codebook is generated, the next stage is to cluster the data. By default the **iteration** count is at 16. Click on '**cluster**'. You will be presented with a real-time preview of the data evolving. If there are still changes in the image noticed, you may click on the **cluster** button again until less changes are apparent
7. Next stage is the c64 output format. The '**dither**' section turns on dithering (recommended). More detailed descriptions of the options later in this document. Experiment until you are satisfied with the results (on the c64 preview bottom left)
8. Click on '**c64 reindex'**. This will re-encode and optimize the data to c64 format.
9. You may now select the '**save c64 codebook'** to save the codebook data and then the '**save c64 vqimage stream'** to save the raw uncompressed data (each frame in 1k chunks merged)

BASIC TUTORIAL (DELTA COMPRESSOR SECTION)



After the image is re-indexed using the 'c64 reindex' button, you may also click on the '**compress**' button in order to invoke the delta compression module of the application. (The delta compression will use the frames / amount of frames specified prior to clicking on the 'c64 reindex button'

You may click on the next/previous frames in order to have a preview on the right on which 8x8 blocks change in each frame. Blue signifies the contents that are the same as the previous frame.

On the right hand side is a live update on the size of each compressed frame as well as the total size of each compressed frame combined.

You may select the compression methods on the left side of the screen (more on this later). Clicking on save data allows you to specify a file name and each compressed frame will be saved separately with the addition of the frame numbers alongside each file.

By all means the user is not forced to use any of these compression methods as the main section of the application allows the raw uncompressed data to be saved allowing the programmer to compress each frame/multiple frames using any method as they wish.

OPTIONS IN DETAIL

FILE MENU

LOAD AVI

This allows the user to load AVI video files directly into the encoder. If there are any issues with loading any AVI file, either install a Codec pack or convert the AVI into an uncompressed file. The file will be loaded using the start frame/amount of frames specified. All frames will be converted to greyscale

LOAD BITMAP

Loads an individual bitmap into the specified frame. Useful if converting one image/a few images. Yet again this is converted to greyscale

LOAD CODEBOOK

Loads a previously saved codebook (in bitmap format) This may be required if the codebook generation took a long time (e.g. using exhaustive search) and to prevent recalculation in the future.

SAVE CODEBOOK

Saves the current codebook in BMP format. Useful for the reason of loading a previously calculated codebook if need be

SAVE BITMAP

Saves the current bitmap (The converted bitmap on the top right hand corner) which is VQ encoded but in 8 bit greyscale

SAVE C64 BITMAP

Saves the current C64 converted bitmap (in bitmap format) this is VQ encoded yet again but converted to the C64's colours

SAVE BITMAP STREAM

Saves the stream of bitmaps (specified via start frame and amount of frames) in individual BMP images. This is yet again VQ encoded but in 8 bit greyscale. This may be of use if wishing to use other external programs to quantise the colours of the image etc and to re-import them into this application

SAVE C64 BITMAP STREAM

Same as the above except that the C64 preview stream is saved

SAVE C64 CODEBOOK

this option is only active once image has been remapped using the 'c64 re-index' option. This will save the codebook (in c64 format) Exactly 2048 bytes (or 4096 bytes) if using the interlaced option (two codebooks)

SAVE C64 VQ IMAGE

This option is only active once image has been remapped using the 'c64 reindex' option. This will save the lookup table frame data which is in c64 format and 1000 bytes in length

SAVE C64 VQIMAGE STREAM

This option is only active once image has been remapped using the 'c64 reindex' option. This will save the merged lookup table frame data specified via start frame and amount of frames with each frame being separated by exactly 1k (1024 bytes)

EXIT

Exits the Application

ABOUT

Displays build date of application (including a rather unflattering pic of me!)

APPLICATION OPTIONS

MSTATIC/PRUNE VALUE

This option has two functions depending on which button is pressed. Both the 'MotionStatic' and 'BSM/Prune' buttons take the data from this value. Of course you would need to change the values depending on which button you are pressing. The data is a tolerance value. Lower amount gives less 'weight' to either the motionstatic or prune routine

CODEBOOK AMOUNT

This should be kept at 256 if using the Prune codebook generation. After all an approximate amount of codebooks required can be generated by increasing the prune amount till the codebooks fill a specific amount of the codebook screen.
If however using the 'Exhaustive search option' then you would need to manually change the amount. (C64 char mode has 256 characters, so not any requirement to change this)

AMOUNT OF FRAMES

The amount of frames that the routines will process. The higher the amount of frames, the longer that the encoding will take. The amount of frames is also used to specify how many frames to load from an AVI file. Maximum of 256 frames with this application.

CURRENT FRAME

You can browse through the frames which the application will automatically decode a preview of how the decompressed image would look like. The main usage for this would be to select the starting point for the encoding / loading of avi/bitmap file.

CLUSTER ITERATION

The amount of times the cluster routine will repeat itself. Usually leave this to 16 and if need be, the user can select the option again. Usually after 32 iterations, the encoded image will not get any better

DITHER STRENGTH

Select the dither strength for the C64 Image conversion. The higher the amount, the more weight there is on the dither. This option is greyed out when the image is C64 re-indexed

BRUTE SEARCH MODE

This turns off prune codebook generation and utilises an exhaustive search to generate the codebook when the 'BSM/Prune' button is pressed. This mode uses a huge amount of RAM and a huge amount of processing time. More on this option later on in the document

RESET

All previously loaded data is stored into a ram buffer. Selecting this option is the equivalent of loading the original files again. Useful when experimenting in the Motion static option

COMPRESS

This option is only available when the data is c64 re-indexed. It transfers the c64 frame data for preparation for compression. More on this later

DITHER

Turn on/off dither. This dithers the data to C64 preview. Recommended to always leave this on. This option is greyed out after c64 re-indexing as dithering data that is already dithered breaks the image. The usage for not dithering the image would be if for example, the image was vq encoded, previews saved, then dithered using another application then re-imported back to this program

POSITIVE DITHER

If the image appears too dark, the dither can be inverted to 'brighten the image'

INTERLACE

Generates two codebooks and interleaves c64 codebook data both in x and y position to reduce flicker immensely. No additional processing time required as the c64 code only needs to change $d018 pointer per frame and move $d016 to a pixel offset each second frame. 320X200 mode in 4+ colours (illusion of more than 4 due to colours overlapping with each other). This option has no affect if 'hires mode' is selected

HIRES

Two colour mode. Can be useful to reduce file size even further as each VQ converted frame will usually have less pixel data which equates to less change of blocks per frame depending on the complexity



Above (in order) Multicolour mode, Hires mode, Interlaced mode.

MOTIONSTATIC

Analyses each frame and keeps 'less moving 8x8 block' sections completely static based on the value in the 'Mstatic/Prune value' box. This greatly increases the quality and compressibility of the frames. However bear in mind that two much of a high amount will destroy the quality completely. This option should only be used right at the beginning (before codebook is generated and before the cluster routine is selected)

BSM/PRUNE

Will either generate a codebook using the prune option or the 'Brute Search Mode' The mode of execution depends on whether the 'brute search mode' box is ticked

CLUSTER

Will cluster each 8x8 block in the image/frames to the nearest codebook entry and merge together to the codebook. This is one of the main parts of the VQ encoding, with the other important part being the codebook generation.

C64 REINDEX

This will re-encode the Converted C64 image as the initial source entry (as well as converting the codebook to C64 format) This will also re-index the VQ frames and remove any unnecessary repeating blocks which had been created due to quantisation.

BRUTE SEARCH MODE

There are various methods of generating a codebook ranging from picking random sections of the image right up to the exhaustive method. The generation of the codebook is one of the most important factors in regards to the final quality of the encoded image.

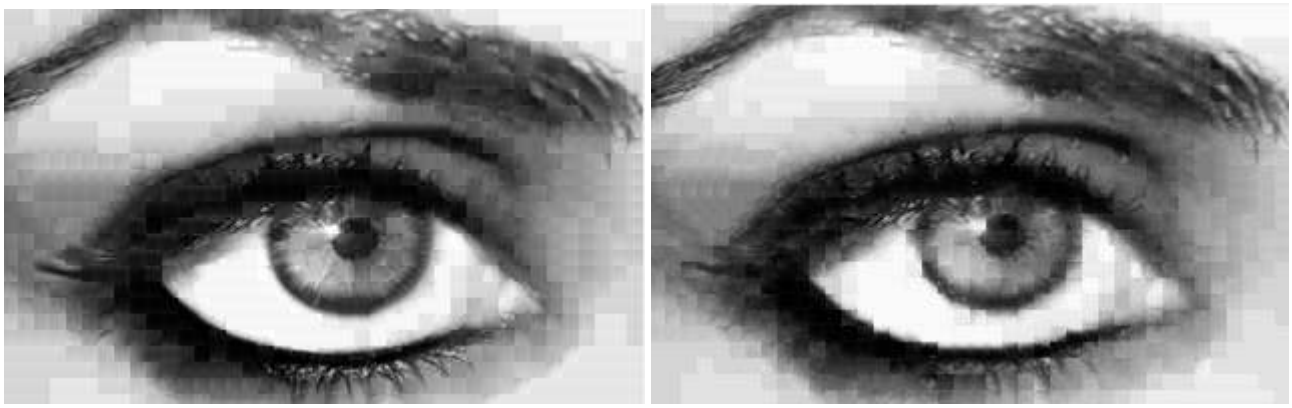
The Brute search option in this application utilises a dynamic shrink/merge routine which generates the same quality as the 'pure exhaustive option' but much much faster. The secret to the speed is only to recalculate what is required and to shrink the size of the error table with each iteration.
However it still runs extremely slowly and memory requirements are immense.

When the 'Brute Search mode' is activated and when the BSM/Prune button is pressed, the routine will first extract only unique blocks and lay them out in a linear table. Bear in mind that just an 8 frame image at 320x200 can have 8000 unique 8x8blocks.

Each block is then compared with each other (every possible combination) and error/index info is saved in a table. Just 8000 frames would require a table of approx (8000x6000) with each table having two entries (error and index lookup) this would equate to around 200-300mb of data

For this reason if the extraction process detects more than 8000 blocks, the routine will not run. If wishing to process more frames using the brute search option, ensure that the source file has less changes between each frame (or simple data) and that a high enough motion static setting is used

Once the routine starts running, it will operate slowly, but increase in speed with each iteration until the codebook is generated.



Above left (Image using BSM) Above right (same image using Prune)

I had other ideas on optimising this, but this did not make it to this application.

Other options to implement would be the split and merge, and ELBG

MOTION STATIC

Many videos consist of static sections but however these would be re-encoded with additional blocks as the data would waver slightly.

The motion static option allows these sections to remain totally still. This increases compressibility via delta encoding as well as ensuring that fewer blocks are required which in turn can increase the quality

DELTA ENCODING COMPRESSION

Once the c64 data is re-indexed, the user can save the uncompressed stream using the file requester, alternatively, the user can click on the '**compress**' button in order to invoke the delta compression module of the application. Brief description of the compression methods as below

XY DELTA

This routine saves the differences of the data by saving the x and y positions of the block that has changed along with the amount of consecutive different blocks after. The X,Y and block transfer amount is saved in two bytes. Following bit structure of the two bytes as follows

XXXXXXRR YYYYYRRR

X = 6 bits (containing 0-39 x pos)
Y = 5 bits (containing 0-24 y pos)

R=bits are merged to form 5 bits (0-31) consecutive data amount.

This is followed by the amount of bytes to copy and this is repeated for each frame with an end of frame marker. This option is only optimal on images with very few changes

DBRLE

Sub categories in this compression method as follows..

X2 Tile BRLE

If each consecutive 2 bytes are empty (delta from previous frame), this is plotted to the tile bit buffer as a setbit otherwise a clearbit.
if they are not the same, then the current char is compared to the first entry, if this is the same, then a setbit in the brle buffer, otherwise a clearbit
if a clearbit, then the new byte is plotted and this becomes the current char for comparison

X4 Tile BRLE

Same as above but operates in 4 values. Usually the most optimum

X8 Tile BRLE

Same as above but operates in 8 values.

Uncomp data

The delta is stored as 'byte 255'. For this reason, the codebook generation does not occupy byte 255. Does not compress the data. May be useful if wishing to experiment and create your own delta compressor

SOURCE CODE (C64)

Examples of c64 decoder may follow. This is not a requirement however as the application can save in RAW VQ streams allowing the user to compress in batch using LZ compression or other methods

KNOWN BUGS

Reindex and Import to delta module not threaded. This may cause the application to freeze. The application has not crashed, be patient and it will be active again (To implement this as a separate thread)

The delta module screen does not always appear infrond of the main application.

XY Comp not tested yet

HISTORY OF CSAM

The first release of CSAM was in 2006/2007 and my Sabrina demo was based on this. This program used an extremely crude method of 'VQ' which should not have really been considered as VQ as there were no mutations/modifications to the tile at all.

After this stage, I had created a few demo's in 2009 and most recent one in 2011 (Algodancer /Algodancer 2) which utilised some of my experimental encoders. Part of this was released as CSAM V2 which was heavily unfinished with many sections missing/not implemented. The addition in CSAM V2 was a fully working Vector Quantiser utilising Cluster merging.

I had a lot of partially unfinished encoders / precompressors and decided to merge them together and the result of this is CSAM V3.

Many other features to implement however as follows

Split Merge/ELBG
Colour (MUCSU)
Manual Selection
Additional compression methods
Self running C64 executables

These exist as separate encoders in the meantime (although not released)

CONTACT

For comments and criticisms, please do not hesitate to contact me at the following

Email algorithm@algotechproductions.com or thealgorithm@hotmail.com

CSDB You can PM me there

Facebook Do a search for 'Naveed Algorithm Khugiani'