A video processing method and concomitant computer software stored on a computer-readable medium comprising receiving a video stream comprising a plurality of frames, removing via one or more GPU operations a plurality of artifacts from the video stream, outputting the video stream with the removed artifacts, and tracking artifacts between an adjacent subset of the plurality of frames prior to the removing step.
FIG. 3A

FIG. 3B

FIG. 4
TEMPORAL MEDIAN CALCULATION STEPS

$V_{ab}$ (Motion vectors between frames 1 and 2)
$N$ (Number of frames in filter)

**Step 1.**
- Frame a
  - $X, Y$
- Frame b
  - $X, Y$
- Frame c
  - $X, Y$

**Step 2.**
- Frame a
  - $X, Y$
- Frame b
  - $X, Y$
- Frame c
  - $X, Y$

**Step 3.**
- Frame a
  - $X, Y$
- Frame b
  - $X, Y$
- Frame c
  - $X, Y$

$V_{ab} = (X_{2} - X_{1}, Y_{2} - Y_{1})$
$V_{cb} = (X_{3} - X_{2}, Y_{3} - Y_{2})$

$\{a', b', c'\} = \text{Median}$

**FIG. 7**
METHOD, APPARATUS, AND COMPUTER SOFTWARE FOR MODIFYING MOVING IMAGES VIA MOTION COMPENSATION VECTORS, DEGRAIN/DENOISE, AND SUPERRESOLUTION

CROSS-REFERENCE TO RELATED APPLICATIONS


[0002] A related application entitled “Method, Apparatus, and Computer Software for Digital Video Scan Rate Conversions with Minimization of Artifacts” is being filed concurrently herewith, to the same Applicants, Attorney Docket No. 31957-Util-3, and the specification and claims thereof are incorporated herein by reference.


STATEMENT REGARDING FEDERALLY SPONSORED RESEARCH OR DEVELOPMENT

[0004] Not Applicable.

INCORPORATION BY REFERENCE OF MATERIAL SUBMITTED ON A COMPACT DISC

[0005] Not Applicable.

COPYRIGHTED MATERIAL

[0006] © 2007-2009 Cinnafilm, Inc. A portion of the disclosure of this patent document and of the related applications listed above contain material that is subject to copyright protection. The owner has no objection to making facsimile reproduction by anyone of the patent document or the patent disclosure, as it appears in the Patent and Trademark Office patent file or records, but otherwise reserves all copyrights whatsoever.

BACKGROUND OF THE INVENTION


[0008] The present invention relates to methods, apparatuses, and software for substantially removing artifacts from motion picture footage, such as film grain and noise effects, including impulsive noise such as dust.

[0009] 2. Description of Related Art:

[0010] Note that the following discussion may refer to publications as to which, due to recent publication dates, are not to be considered as prior art vis-à-vis the present invention. Discussion of such publications herein is given for more complete background and is not to be construed as an admission that such publications are prior art for patentability determination purposes. The need and desire to make video, particularly that converted from stock footage on traditional film, look less grainy and noisy is a considerable challenge due to high transfer costs and limitations of available technologies that are not only time consuming, but provide poor results. The present invention has approached the problem in unique ways, resulting in the creation of a method, apparatus, and software that not only changes the appearance of video footage to substantially remove film grain and noise effects, but performs this operation in real-time or near real-time. The invention (occasionally referred to as Cinnafilm®) streamlines current production processes for professional producers, editors, and filmmakers who use digital video to create their media projects. The invention permits conversion of old film stock to digital formats without the need for long rendering times and extensive operator intervention associated with current technologies.

BRIEF SUMMARY OF THE INVENTION

[0011] The present invention is of a video processing method and concomitant computer software stored on a computer-readable medium, comprising: receiving a video stream comprising a plurality of frames; removing via one or more GPU operations a plurality of artifacts from the video stream; outputting the video stream with the removed artifacts; and tracking artifacts between an adjacent subset of the plurality of frames prior to the removing step. In the preferred embodiment, tracking comprises computing motion vectors for the tracked artifacts, including computing motion vectors for the tracked artifacts with at least a primary vector field and a secondary vector field with double the resolution of the primary vector field, computing motion vectors for the tracked artifacts via subpixel interpolation without favoring integer pixel lengths, and/or computing motion vectors for the tracked artifacts with a hierarchical set of resolutions of frames of the video stream. Removing comprises removing artifacts that are identified via assumption that a motion compensated image signal is relatively constant compared to the artifacts, including employing a temporal wavelet filter by motion compensating a plurality of frames to be at a same point in time, performing an undecimated wavelet transform of each temporal frame, and applying a filter to each band of the wavelet transform and/or employing a Wiener filter using as an input a film grain profile image sequence extracted from the plurality of frames to remove film grain artifacts. Artifacts are prevented from being introduced into the video stream via a motion compensated temporal median filter employing confidence values. Superresolution analysis is performed on the video stream that is constant in time with respect to a number of frames used in the analysis.

[0012] Objects, advantages and novel features, and further scope of applicability of the present invention will be set forth in part in the detailed description to follow, taken in conjunction with the accompanying drawings, and in part will become apparent to those skilled in the art upon examination of the following, or may be learned by practice of the invention. The objects and advantages of the invention may be
realized and attained by means of the instrumentalities and combinations particularly pointed out in the appended claims.

BRIEF DESCRIPTION OF THE SEVERAL VIEWS OF THE DRAWINGS

[0013] The accompanying drawings, which are incorporated into and form a part of the specification, illustrate one or more embodiments of the present invention and, together with the description, serve to explain the principles of the invention. The drawings are only for the purpose of illustrating one or more preferred embodiments of the invention and are not to be construed as limiting the invention. In the drawings:

[0014] FIG. 1 illustrates an inefficient standard implementation of transforming S into HH, HL, LH, and LL subbands of the wavelet transform;

[0015] FIG. 2 illustrates a preferred efficient GPU implementation using multiple render targets;

[0016] FIGS. 3(a) and 3(b) are diagrams illustrating colored vector candidates in M for the corresponding motion vectors in field M+1; dashed vectors identify improvements in accuracy along the edge of an object;

[0017] FIG. 4 is an illustration of subpixel interpolation; the original motion vector is dotted, and the shifted vector to compensate for subpixel interpolation is solid; bold grid lines are integer pixel locations, lighter grid lines are fractional pixel locations; in this example, the original vector had interpolation factors of (0,0) → (0.75, 0.5); the adjusted vector has interpolation factors of (0.125, 0.75) → (0.875, 0.25), both of which are equally distant from the nearest of 0 or 1;

[0018] FIGS. 5(a) and 5(b) show an example of chunk filtering, with 50% overlap, M=4N=4x4, processed in four chunks of 2x2 blocks each;

[0019] FIGS. 6(a)–6(c) illustrate a technique of selecting candidate block sets according to the invention, with Q=½ for the first three frames; after the first three frames the pattern repeats; a shaded square indicates a block in the grid selected to be a candidate block; and

[0020] FIG. 7 is an illustration of the preferred temporal median calculation steps of the invention.

DETAILED DESCRIPTION OF THE INVENTION

[0021] Embodiments of the present invention relate to methods, apparatuses, and software to enhance moving video images at the coded level to remove (and/or add) artifacts (such as film grain and other noise), preferably in real time (processing speed equal to or greater than ~30 frames per second). Accordingly, with the invention processed digital video can be viewed "live" as the source video is fed in. So, for example, the invention is useful with video "streamed" from the Internet, as well as in converting motion pictures stored on physical film.

[0022] Although the invention can be implemented on a variety of computer hardware/software platforms, including software stored in a computer-readable medium, one embodiment of hardware according to the invention is a stand-alone device, which is next described. Internal Video Processing Hardware preferably comprises a general purpose CPU (Pentium4® or Core2 Duo®, Core2 Quad® class), graphics card (DX9 or better capable), system board with expandability for video I/O cards (preferably PCI compatible), system memory, power supply, and hard drive. A Front Panel User Interface preferably comprises a standard keyboard and mouse usable menu for access to image-modification features of the invention, along with three dials to assist in the fine tuning of the input levels. The menu is most preferably displayed on a standard video monitor. With the menu, the user can access at least some features and more preferably the entire set of features at any time, and can adjust subsets of those features. The invention can also or alternatively be implemented with a panel display that includes a touchscreen.

[0023] The apparatus of the invention is preferably built into a sturdy, thermally proficient mechanical chassis, and conforms to common industry rack-mount standards. The apparatus preferably has two sturdy handles for ease of installation. I/O ports are preferably located in the front of the device on opposite ends. Power on/off is preferably located in the front of the device, in addition to all user interfaces and removable storage devices (e.g., DVD drives, CD-ROM drives, USB inputs, Firewire inputs, and the like). The power cord preferably extrudes from the unit in the rear. An Ethernet port is preferably located anywhere on the box for convenience, but hidden using a removable panel. The box is preferably anodized black wherever possible, and constructed in such a manner as to cool itself via convection only. The apparatus of the invention is preferably locked down and secured to prevent tampering.

[0024] An apparatus according to a non-limiting embodiment of the invention takes in a digital video/audio stream on an input port (preferably SDI, or from a video data file or files), and optionally uses a digital video compression-decompression software module (CODEC) to decompress video frames and the audio buffers to separate paths (channels). The video is preferably decompressed to a two dimensional (2D) array of pixel interleaved luminance-chrominance (YCbCr) data in either 4:4:4 or 4:2:2 sampling, or, optionally, red, green, and blue color components (RGB image, 8-bits per component). Due to texture resource alignment requirements for some graphics cards, the RGB image is optionally converted to a red, green, blue, and alpha component (RGBA, 8-bits per component) buffer. The audio and video is then processed by a sequence of operations, and then can be output to a second output port (SDI) or video data file or files.

[0025] Although other computer platforms can be used, one embodiment of the present invention preferably utilizes commodity x86 platform hardware, high end graphics hardware, and highly pipelined, buffered, and optimized software to achieve the process in real-time (or near real-time with advanced processing). This configuration is highly reconfigurable, can rapidly adopt new video standards, and leverages the rapid advances occurring in the graphics hardware industry.

[0026] In an embodiment of the present invention, the video processing methods can work with any uncompressed video frame (YCbCr or RGB 2D array) that is interlaced or non-interlaced and at any frame rate, including 50 or 60 fields per second interlaced (50i, 60i), 25 or 30 frames per second progressive (25p, 30p), and 24 frames per second progressive, optionally encoded in the 2:3 pulldown or 2:3:3:2 pulldown formats. In addition to DV, there are numerous CODECs that exist to convert compressed video to uncompressed YCbCr or RGB 2D array frames. This embodiment of the present invention will work with any of these CODECs.

[0027] The present application next describes the preferred methods employed by the invention. For purposes of the specification and claims, an 'operation' is the fundamental
A building block of the Cinnafilm engine. An operation has one critical function, ‘Frame’, which has the index of the frame to be processed as an argument. The operation then queries upstream operations until an input operation is reached, which implements ‘Frame’ in isolation by reading them from an outside source (instead of processing existing frames).

There are preferably four types of operations: (1) Video operations, (2) GPU (Graphics Processing Unit) operations, (3) Audio operations, and (4) Interleaved operations. The type of operation indicates what type of frame that operation operates on. Interleaved frames are frames that possess both a video and an audio frame. GPU frames are video frames that are stored in video memory on a graphics card. GPU operations transform one video memory frame into another video memory frame.

A few key operations bridge frames between frame types: (1) GPU converts video to GPU frames and back. It is technically a video operation, but it accepts GPU operations as its child nodes. Video frames go into the GPU operation, are processed by GPU operations on the GPU, and then the GPU operation downloads the frames back to the CPU for further processing. (2) AudioVideo converts interleaved frames into separate audio and video frames which can be processed by audio and video operations.

A few preferred operations are described herein: (1) TemporalMedian; (2) WaveletTemporal; (3) NoiseFilter; (4) WaveletFilter; (5) WienerFilter; (6) WienerAnalysis; (7) SuperResolution; and (8) BlockMatch.

The present application next describes the preferred GPU implementation details of the Fast Fourier Transform using Cooley Tukey and Stockham auto-sort. The 2D (two-dimensional) Fourier transform is performed on the GPU in six steps: Load, transform X, transform Y, inverse transform X, inverse transform Y, and Save. After the forward transform (first three steps), but before the inverse transform (last three steps), any number of filters and frequency domain operations can be applied. Each group of six steps plus the filtering operations operates on a number of variously overlapping blocks in the input frame. The load operation handles any windowing, zero padding and block overlapping necessary in order to make the frame fit into a set of uniformly sized blocks that is a power of two. Once loaded, transform X and transform Y are performed with the Stockham auto sort algorithm for performing Cooley Tukey decimation-in-time. These two steps are identical except for the axis on which they operate. Inverse transform in X and Y is performed by using the same algorithm as transform X and transform Y except for negating the sign of the twiddle factors and a normalization constant. Once the transforms are inverted, the save operation uses the graphics card’s geometry and alpha blending capability to overlap and sum the blocks, again with a weighted window function. This is accomplished by drawing the blocks as a set of overlapping quads. Alternatively, a shader program can be employed to compute the addresses of the pixels within the required blocks.

Fourier transforms on the GPU are performed with two complex transforms in parallel vector-wise. Two complex numbers x+iy and x'+iy' are stored in a 4 component vector as (x1, y1, x2, y2). GPUs typically operate most efficiently on four component vectors due to their design for handling RGBA data. Then, many transforms are performed in parallel by putting many blocks into a single texture. For example, a frame broken up into M blocks x N blocks would be processed in one call by putting M x N blocks in a single texture. The parallelism is realized by having many instances of the Fourier transform program processing all of the blocks at once. The more blocks, and by extension more image pixels or frequency bins available to the GPU, the more effectively the GPU will be able to parallelize its operations.

A set of analysis (forward Fourier transform) and synthesis (inverse Fourier transform) window functions suitable for manipulation by an untrained user can be defined by Weighted Hann(x, w) = |Hann(x)| w. The present invention provides an adjustable window function from 0 to 1; the analysis window function is then defined to be Weighted Hann(x, W), and the synthesis window function is defined to be Weighted Hann(x, 1-W). This provides the ability for user adjustability of the frequency domain algorithms without requiring advanced knowledge.

The present invention also provides an efficient implementation of Discrete Wavelet Transforms on the GPU, preferably following techniques disclosed in Starck, et al., “The Undecimated Wavelet Decomposition and its Reconstruction”, IEEE Transactions on Image Processing 16:2, February 2007, pp. 297-309. Several features of modern GPUs allow for efficient implementations of the Discrete Wavelet Transform. In particular, as shown in the distinction between FIGS. 1 and 2, multiple render targets allow for the computation of all 4 sub-bands (HH, HL, LH, LL) of a given level of the transform from scaling coefficients (S) in one pass, whereas the standard implementations or implementations on other hardware may require 4 independent passes to produce each sub-band. This reduces the memory bandwidth used for input data by a factor of 4. This applies at least to undecimated wavelet transforms, and can be applied to decimated wavelet transforms.

The present invention preferably employs improvements to motion estimation algorithms, including those disclosed in the related applications.

First, the invention provides a method for efficiently improving resolution of motion vector field, as follows: Suppose that sufficiently accurate motion vectors have been determined and are stored in a motion vector field M. To inexpensively improve the resolution of these accurate motion vectors, consider a new motion vector field M+1 with double the resolution of M. Each vector in M+1 has only four candidate vectors: The nearest four vectors in M, as shown in FIGS. 3(a) and 3(b). The reasoning is that if a block straddles a border of motion, the block must choose one of the areas of motion to represent. However, in a further subdivided level, the blocks may fall entirely in one region of motion or the other, which differs from the choice of the coarsest block. One of the four neighbors of the coarsest block should be the correct vector because one of the neighbors lies entirely within the area of motion which this new subdivided block entirely belongs to as well. This candidate vector should be the result of the motion estimation optimization. Note that this technique will never produce new vectors, so it is only suitable for refinement after coarse but accurate motion vectors are found. This technique vastly improves motion vector accuracy near sharp edges in the original image.

Second, the invention provides a method for improving accuracy of subpixel accurate motion estimation. A problem that is encountered when attempting to perform subpixel accurate motion estimation is that when evaluating a candidate vector, the origin of the vector is placed at an integer pixel location, while the end of the vector can end up
at a fractional pixel location, requiring subsampling of the image. This problem will be referred to as unbalanced interpolation.

[0038] Suppose the current block whose displacement is being estimated is located at point \( x \), and the displaced location is \( x' \). \( x \) has an integer pixel location, while \( x' \) may have a sub-pixel component. When sampling the images \( A \) at point \( x \) and \( B \) at \( x' \), \( A \) is sampled on whole pixel locations, while \( B \) is interpolated according to the sub-pixel component of \( v \). This results in a sub-optimal matching process because unequal amounts of interpolation are applied which favors integer pixel vector lengths.

[0039] Bellers, et al., “Sub-pixel accurate motion estimation”, Proceedings of SPIE, VCIP’99, January 1999, pp. 1452-1463, suggest using Catmull-Rom cubic interpolation or even higher complexity filters to avoid the same issue of unbalanced image interpolation. However, interpolation of the image data is in the very innermost loop of the motion estimator, and using cubic interpolation instead of linear is an extremely heavy price to pay in performance. To avoid this penalty in performance, some implementations upsample the input images before beginning the motion estimation process. This still has a large performance penalty due to increasing the memory usage and bandwidth by the motion estimation process.

[0040] The preferred method of the invention solves the same problem, but with a relatively cheap operation, as follows.

[0041] The inventive solution to this problem, as illustrated in FIG. 4, is to displace both \( x \) and \( x' \) by a carefully computed value that represents the sub pixel component of \( v \). Let \( v = v + v_y \), and \( v = v - v_y \), \( v_y \) represents the sub pixel component of \( v \) while \( v \) is the integer component. Let \( v = v_y \), \( v = v_y / 2 \), the adjusted motion vector. Then displace \( x \) by \( v \) to find \( x' \), and also displace \( x \) by \( v_y / 2 \). This results in both \( x \) and \( x' \) containing equal sub pixel components, while not significantly affecting the actual position of the motion vector (a vector will never be moved more than \( 1/4 \) of a pixel in either axis since \( \text{abs}(v_y/2) \leq 1/4 \)). This technique results in less error introduced by interpolation when computing sub pixel accurate motion vectors.

[0042] Third, the invention provides a method for improving motion estimator robustness against image noise in hierarchical block matching motion estimation algorithms. Standard block matching algorithms work by forming a candidate vector set \( C \). Then, each vector in the set is scored by computing the difference in pixels \( d \) the vector to be applied to the images. In areas of low image information (low signal-to-noise ratio) the motion vector data can be very noisy doing to the influence of noise in the image. This noise reduces the precision and reliability of some algorithms relying on motion compensation; for example, temporal noise filtering or cut detection. Therefore, it is important to combat this noise produced by the algorithm.

[0043] The inventive method applies to hierarchical motion estimators as follows. The first step is to form an image resolution pyramid, where the bottom of the pyramid is the full resolution image, and the subsequent levels are repeatedly downsampled by a factor of two. Motion estimation begins by performing the search described above of the immediate neighbors at the top of the pyramid, and feeding the results to the subsequent levels which incrementally increase the accuracy of the motion vector data. At each level, the candidate vector set is the immediately neighboring pixel in every direction.

[0044] To improve motion estimator robustness against image noise, the invention defines a constant \( e \). When optimizing the candidate vector set \( C \) and the current best vector \( v \) taken from the previous level in the hierarchy, define \( e \) to be the minimally scoring vector candidate. The standard behavior is to select \( \min \{ e, v \} \). The inventive solution against noise is to select \( \min \{ e, e + v \} \). This way, a vector only is selected if the candidate vector is decisively better than the current vector (from the previous level). This results in the existing standard behavior in areas of detail (high SNR), where motion vectors can reliably be determined, and in areas of low detail (low SNR) the vectors are not noisy.

[0045] Since the noise is inherently reduced in higher levels of the image pyramid by filtering, \( e \) is adjusted per level of the hierarchy to be small for the highly filtered levels of the image pyramid and large at the lowest level. In our implementation, \( e = e^{(l+1)} \), where \( e \) is a user defined parameter, \( e \) is the constant value for level \( l \) and \( l = 0 \) is the highest resolution level of the image pyramid. This has been empirically determined to perform better than other methods such as \( e^{(l+1)} \).

[0046] The invention also provides for reducing noise and grain in moving images using motion compensation. The inventive method exploits the fact that the motion compensated image signal is relatively constant compared to random additive noise. Film grain is additive noise when the data is measured in film density. Before processing film grain stored in log density space (very commonly used in DPX file format for example), the data should preferably be transformed to be in linear density space, resulting in the grain being additive noise.

[0047] The Temporal Wavelet filter of the invention works by motion compensating several frames (TEMPORAL_SIZE frames) to be at the same point in time, performing the undecimated wavelet transform of each temporal frame, and by applying a filter to each band of the wavelet transform. Additionally, the scale coefficients of the lowest level of the wavelet transform are also preferably temporally filtered to reduce the lowest frequency noise. Two operations implement temporal wavelet filtering: Wavelet Temporal and Noise Filter.

[0048] Each filter starts by collecting TEMPORAL_SIZE frames surrounding the frame to be filtered. This forms the input set. The frames in the input set are motion compensated to align with the output frame (temporally in the center of the input set). Then an undecimated wavelet transform is applied using the above-described efficient implementation of discrete wavelet transforms on the GPU, using an appropriate set of low and high pass filters. For example, one possible set of filters is [121]/4 (a three-tap Gaussian filter) the low pass filter, and the high pass filter is [010]–[121]/4 (a delta function minus a three-tap Gaussian). The undecimated wavelet transform is performed using the “a trous” algorithm.

[0049] In the wavelet domain, the detail coefficients are filtered preferably using a filtering method robust against motion compensation artifacts. For Noise Filter, the detail coefficients of a one level wavelet transform is filtered using a hierarchical 3D median filter, and the scaling coefficients are filtered using a temporal Wiener filter. For Wavelet Temporal, all coefficients from an optional number of levels are filtered using a temporal Wiener filter. The Wiener filter in this application is robust against motion compensation artifacts.
Using the isotropic wavelet transform results in a significant reduction in memory usage and modest improvement in processing complexity in exchange for a small sacrifice in filter performance.

The invention preferably employs applying a motion compensated Wiener filter to reduce noise in video sequence. The preferred Wiener filter of the invention (compare to U.S. Pat. No. 5,500,685, to Korkoros) uses the Fourier transform operation outlined above. The Wiener filter has several inputs: SPATIAL\_SIZE (block size), TEMPORAL\_SIZE (number of frames), AMOUNT, strength in individual RGB channels, and most importantly, a grain profile image sequence. The grain profile can either be user selected or found with an automatic algorithm. An algorithm is given below with a method to eliminate the need for a sequence.

The grain profile image is a clean sample of the grain, which is at least SPATIAL\_SIZE by SPATIAL\_SIZE pixels, and lasts for TEMPORAL\_SIZE frames. Once the grain profile image is known, the image data is offset to be zero mean, and the 3D Fourier transform is performed to produce a SPATIAL\_SIZE by SPATIAL\_SIZE by TEMPORAL\_SIZE set of frequency bins. The power spectrum is then found from this information. This power spectrum is then uploaded to the graphics card for use within the filter.

The filter step begins by collecting TEMPORAL\_SIZE frames. This forms the input set. These frames are then motion compensated to align image details temporally in the same spatial position. The output frame is the middle frame of the input set, if the set is of even size; the output frame is the later frame of the two middle frames. Once the frames are collected, each one is split into overlapping blocks and the Fourier transform is applied as above. Then, the 3D (three dimensional) Fourier transform is produced by taking the Fourier transform across the temporal bins in each 2D transform. Once the 3D transform is found, then the power spectrum is computed.

Now both the grain profile power spectrum and the image power spectrum is available. The filter gain for the power spectrum bin x, y, i is defined by: G(x, y, i) = F(x, y, i)/F(x, y, i+1)AMOUNT*G(x, y, i), where F is the power spectrum of the video image, and G is the power spectrum of the grain profile image sequence.

AMOUNT is computed to be the overall strength of the filter multiplied with the strength in the current channel being filtered, and is not employed in a typical Wiener filter implementation. These parameters are specified by the user. The default value is AMOUNT=1.0.

Techniques are preferably used to reduce the excessive memory usage demanded by the simple implementation of this filter. First, two image channels are packed into one transform by putting channel 1 in the real component and channel 2 in the imaginary component. Once in the frequency domain, the individual channels are extracted by using linearity and the symmetry property of the Fourier transform of a real sequence. This results in a 50% reduction in memory usage and computations because two transforms are required instead of three. Second, it is necessary to perform the filtering in chunks of the blocks. Referring to FIGS. 5(a) and 5(b), let the image consist of MxN overlapped blocks. The naive implementation could process all MxN blocks at once, and sum the blocks with alpha blending in one set of overlapped geometries as explained above. However, this task can be split up into several sets of blocks, such as [0, M/2) x [0, N/2), [M/2, M) x [0, N/2), [0, M/2) x [N/2, N), [M/2, M) x [N/2, N]. This has reduced the memory usage by 75% (1/4th of the original footprint), because only one of the sets of blocks is required in memory at once. This process can be done at more than just a factor of two. A factor of four would reduce memory usage by 1/16th for example (1/16th of the original footprint). The geometry processing and alpha blending capability of the GPU is exploited to perform overlapped window calculations over multiple passes (one for each chunk of blocks).

The invention further provides for automatically locating a suitable grain profile image for the Wiener filter. To find a suitable profile image, define a cost function as the impact of the filter kernel described above. Therefore, the goal is to minimize G. One preferably minimizes the maximum possible bin of G. Therefore, a suitable grain profile image can be found by computing the power spectrum density of a set of candidate blocks in a frame. Select the block with the minimum maximal power spectrum density as the best block in the frame, and then temporally optimize the best candidate blocks over many frames. The optimization is performed independently in the separate RGB channels, so the final set of optimal R, G, B images may not be from the same location in the same original image.

A significant part of this algorithm is determining the candidate block set. A small candidate block set is important for efficiency purposes. To optimize this candidate block set, observe that in the vast majority of footage, motion is relatively low. This means that a block at some point x, y in one frame is likely very similar to the block at the same x, y in the nearby neighboring frames. This fact is exploited to reduce computational load: split each frame into a grid aligned on the desired block size (SPATIAL\_SIZE in the Wiener filter). A full search would define the candidate block set as every block in this grid. Instead, define a quality parameter Q in (0, 1). A given block in the grid should only be tested in 1/Q frames. To accomplish this, a block is defined to belong to the candidate block set if: x+y+i=mod ceil(1/Q), where x and y are the block coordinates in the grid, and i is the frame index. In this way the computational load is distributed equally across many frames, and provided that the camera motion is low enough for the given Q, every possible sample of grain will be tested. Note that Q=1 corresponds to a full search (the entire grid belongs to the candidate vector set). FIGS. 6(a)-6(c) illustrate Q=1/3. It is preferred that Q=1/4 by default.

The invention also provides for improving usability of selecting profile images. In the above filter description, the power spectrum of the noise is required for the full temporal window. In the standard implementation, this implies a requirement of a sequence of TEMPORAL\_SIZE frames to profile from. In practice, this is difficult to accomplish and places another dimension of constraints on the profile images, which are already difficult to find in real, practical video.

To improve usability, only a single frame of grain profile is required. Then, up to seven more unique profile images can be generated by rotating the original profile three times, then mirroring it, and rotating it another three times. In practice, artifacts resulting from this technique are minimal.

The invention further provides for reducing noise with an intraframe wavelet shrinkage. The preferred filter employs a wavelet based spatial filtering technique, and is derived from profiling a sample of the grain to determine appropriate filter thresholds. The filter thresholds are then adjustable in the user interface with live feedback.
The filter begins by performing the undecimated wavelet transform using three tap Gaussian filters up to some predefined number of levels, presumably enough levels to adequately isolate the detail coefficients responsible for the presence of noise. The preferred implementation is variable up to four levels. The detail coefficients of each level are then thresholded using soft thresholding or another thresholding method.

The filter thresholds are determined by profiling a sample designated by the user to be a mostly uniform region without much detail (a region with low signal to noise ratio). For each level, the filter thresholds are determined using the 1st and 2nd quartiles of the magnitude of the detail coefficients. This statistical analysis was chosen for its property that some image detail can be present in the wavelet transform of the area being profiled without affecting the lower quartiles. Therefore it is robust against user error for selecting inappropriate profiles, or allows for suboptimal profiles to be selected if no ideal profile is available.

In the special case of film grain, it is sometimes necessary to transform the image data into density space (from log density or otherwise) to transform the grain into additive noise.

A significant optimization can be made in processing time and memory usage when transformation is necessary: Instead of transforming the image data (which also invalidates the filter thresholds determined from a profile image in a different basis), the scaling coefficients of the wavelet transform are examined, and the inverse transform is applied to the filter thresholds. The grain is not transformed into additive white Gaussian noise at all, but the filter thresholds adapt to the luminance at the particular location as if the noise were transformed to be additive. Let T be the image data, T be the transformation from log density to density, and F be the filter function. The standard implementation is to process the image as d=F[T(F(T(a))]. The invention is to use d=F^{-1}[T(a)] as a filter function.

The invention also provides for reducing impulsive noise in moving image sequence via an artifact resistant method of motion compensated temporal median. The motion compensated temporal median filter works by first finding forward and backward motion vectors, using Cinmafilm’s block motion estimation engine. Once motion vectors are known, some odd number N of contiguous frames, centered on the frame desired for output, are motion compensated to produce N images of the same frame (the center frame has no motion compensation). Then a median operation is applied to the N samples to produce the output sample.

The temporal median filter is very effective for removing impulsive noise such as dust and dirt in film originated material. Note in FIG. 7 how the large black particle of dust was eliminated from the ball because the median selects the majority color present — red. If the black dot were real image detail, it would have been present in all three frames in the same location, and the median filter would not have filtered it.

The motion estimator produces a confidence value which is used by the temporal median filter to prevent artifacts caused by incorrect motion vectors. In the case of low confidence values, the median filter size is either recursively reduced to N=2, or if N=2−1, a median operation is applied and the original sample is used as the output (no filtering). The invention further provides for a practical technique for performing superresolution on moving images using motion vector solutions. Standard superresolution algorithms work by finding subpixel accurate motion vectors in a sequence of frames, and then using fractional motion vectors to reconstruct pixels that are missing in one frame, but can be found in another. This is a very processor intensive task that can be made practical by keeping a history of the previous frames available for finding the appropriate sampling of a pixel. Let F \in \mathbb{R}^{n \times n} be a sequence of frames at some resolution M\times N. The superresolution image is some resolution M\times N_{3} which is greater than the original resolution. Let S be the superresolution history image, which has resolution M\times N_{3}. This image should have two components, the image color data (for example, RGB or YCbCr), and a second component which is the rating for that pixel. Note that this can be efficiently implemented on standard graphics hardware which typically has 4 channels for a texture: the first three channels store the image data (capable of holding most standard image color spaces), and the fourth stores the rating.

The score value is a rating of how well that pixel matches the pixel at that resolution, where zero is a perfect score. For example, suppose M\times N_{3} is exactly twice M\times N_{1}. Then for the first frame, the superresolution image pixels 0, 2, 4, 6, \ldots \times 0, 2, 4, 6 \ldots should have a perfect score because they are exactly represented by the original image. For pixels not exactly sampled in the original image, they must be found in previous images using motion vector analysis. If a motion vector has a fractional part of 0.5, then it is a perfect match for the odd pixels in the previous example. This is because that pixel in the previous image moved to exactly half way between the two neighboring pixels in the subsequent (current) image. Define the score to be some norm (length, squared length, etc.) of the difference of the vector’s fractional part from the ideal. In this case, 0.5 is the ideal, and if the motion vector has a fractional part of 0.5, then it is a perfect match and the score is 0.

When any new frame is processed, the image S is updated such that each pixel is the minimum score of either the current value, or the new value. To prevent error from accumulating, the scores of S are always incremented by some decay value. This prevents a perfect match from persisting for too long, and favors temporally closer frames in the case of near ties.

Using this technique, superresolution analysis becomes a constant time algorithm with respect to the number of frames used in the analysis. The number of frames used is controlled by the decay parameter. High values of decay mean a smaller number of frames will be used to search for the best subsample match. However, this algorithm demands more accuracy from the motion estimation algorithm due to the potential for error to accumulate.

Preferably, the invention employs a multpass improvement in superresolution algorithm quality. Once the above technique is established, accuracy can be improved by a multpass method. First perform the above algorithm has described, and then perform the algorithm again on the input frames in reverse order. In both passes, the complete S image, i.e., including the score values should be stored for each frame. After both passes are complete, a third pass is performed which minimizes the score values from each pass. This results in a complete neighborhood of frames being analyzed and used for the superresolution algorithm results, as opposed to only frames in one direction as in a single pass.
[0074] Note that in the specification and claims, “about” or “approximately” means within twenty percent (20%) of the numerical amount cited.

[0075] Although the invention has been described in detail with particular reference to these preferred embodiments, other embodiments can achieve the same results. Variations and modifications of the present invention will be obvious to those skilled in the art and it is intended to cover in the appended claims all such modifications and equivalents. The entire disclosures of all references, applications, patents, and publications cited above are hereby incorporated by reference.

What is claimed is:

1. A video processing method comprising the steps of:
receiving a video stream comprising a plurality of frames;
removing via one or more GPU operations a plurality of artifacts from the video stream;
outputting the video stream with the removed artifacts; and
tracking artifacts between an adjacent subset of the plurality of frames prior to the removing step.

2. The method of claim 1 wherein the tracking step comprises computing motion vectors for the tracked artifacts.

3. The method of claim 2 wherein the tracking step comprises computing motion vectors for the tracked artifacts with at least a primary vector field and a secondary vector field with double the resolution of the primary vector field.

4. The method of claim 2 wherein the tracking step comprises computing motion vectors for the tracked artifacts via subpixel interpolation without favoring integer pixel lengths.

5. The method of claim 2 wherein the tracking step comprises computing motion vectors for the tracked artifacts with a hierarchical set of resolutions of frames of the video stream.

6. The method of claim 1 wherein the removing step comprises removing artifacts that are identified via assumption that a motion compensated image signal is relatively constant compared to the artifacts.

7. The method of claim 6 wherein the removing step comprises employing a temporal wavelet filter by motion compensating a plurality of frames to be at a same point in time, performing an undecimated wavelet transform of each temporal frame, and applying a filter to each band of the wavelet transform.

8. The method of claim 6 wherein the removing step comprises employing a Wiener filter using as an input a film grain profile image sequence extracted from the plurality of frames to remove film grain artifacts.

9. The method of claim 1 additionally comprising the step of preventing artifacts being introduced into the video stream via a motion compensated temporal median filter employing confidence values.

10. The method of claim 1 additionally comprising the step of performing superresolution analysis on the video stream that is constant in time with respect to a number of frames used in the analysis.

11. Computer software stored on a computer-readable medium for manipulating a video stream, said software comprising:
software accessing an input buffer into which at least a portion of said video stream is at least temporarily stored; and
software removing via one or more GPU operations a plurality of artifacts from at least a portion of said video stream; and
wherein via tracking software artifacts are tracked between an adjacent subset of the plurality of frames prior to execution of the removing software.

12. The software of claim 11 wherein the tracking software comprises software computing motion vectors for the tracked artifacts.

13. The software of claim 12 wherein the tracking software comprises software computing motion vectors for the tracked artifacts with at least a primary vector field and a secondary vector field with double the resolution of the primary vector field.

14. The software of claim 12 wherein the tracking software comprises software computing motion vectors for the tracked artifacts via subpixel interpolation without favoring integer pixel lengths.

15. The software of claim 12 wherein the tracking software comprises software computing motion vectors for the tracked artifacts with a hierarchical set of resolutions of frames of the video stream.

16. The software of claim 11 wherein the removing software comprises software removing artifacts that are identified via assumption that a motion compensated image signal is relatively constant compared to the artifacts.

17. The software of claim 16 wherein the removing software comprises software employing a temporal wavelet filter by motion compensating a plurality of frames to be at a same point in time, performing an undecimated wavelet transform of each temporal frame, and applying a filter to each band of the wavelet transform.

18. The software of claim 16 wherein the removing software comprises software employing a Wiener filter using as an input a film grain profile image sequence extracted from the plurality of frames to remove film grain artifacts.

19. The software of claim 11 additionally comprising software preventing artifacts being introduced into the video stream via a motion compensated temporal median filter employing confidence values.

20. The software of claim 11 additionally comprising software performing superresolution analysis on the video stream that is constant in time with respect to a number of frames used in the analysis.