Super-resolution Frame Reconstruction Using Subpixel Motion Estimation

ABSTRACT

When video data is used for forensic analysis, it may transpire that the level of detail available is insufficient. This is particularly the case when it is derived from sensors deployed in the field since they have limited resolution. For example, a vehicle license plate number may not be legible in video captured during a traffic violation.

In such cases, it is desirable to enhance the resolution of selected frames. However, obtaining the maximum resolution possible is computationally prohibitive. Forensic analysts need a tool where the tradeoff between detail extracted and the running time can be controlled.

We describe a parametrized algorithm for obtaining higher resolution frames using temporal context in the video stream. By varying the framework's variables, the quality of the output and the cost of computing it can be traded. We briefly report on our proof of concept implementation as well.

1. INTRODUCTION

Video is routinely used to record events for analysis at a later time. Surveillance cameras are used to monitor a range of locations, including product shelves in shops, traffic on roads, entrances to buildings, and users of ATMs (automatic teller machines). When a crime occurs, the recorded footage may be used to identify the perpetrator.

Since the cost of a video sensor is directly related to the resolution at which it captures single frames, the size chosen is usually a compromise between cost and function. As a result, when the data is being used for forensic applications, it may be necessary to enhance its resolution. For example, this may be the only way of reading the license plate on a passing car.

The cost of using temporal context to improve the resolution of a single frame is significant. Hence, a forensic analyst would typically prefer to select the extent of improvement in resolution that is required, since this would also control how much time the process would take. Our algorithm and prototype implementation provide this.

1.1. Background

In the context of digital video, an approximation occurs during the recording process. The image space in $\mathbf{R}^2$ is approximated with a *representation grid* in $\mathbf{N}^2$, which we call the *approximated image space*. Thus, regions in $\mathbf{R}^2$ are represented by a single data point, a *pixel*, as illustrated in Figure 1.

By repeating the spatial registration process at discrete points in time, a *video sequence* is formed. Recording the instant in time with a temporal index is accomplished by a strictly increasing monotonic mapping from $\mathbf{R}$ to $\mathbf{N}$.

Finally, the set of points that undergoes spatial projection will vary from *frame* to frame. Video compression algorithms rely on the eye's inabaility to discern small variations. For example, quantization maps a range of intensity levels to a single one since it can be represented more succinctly. Our goal is to extract and utilize this variation instead. It will provide us with extra information about regions in temporally proximal frames.

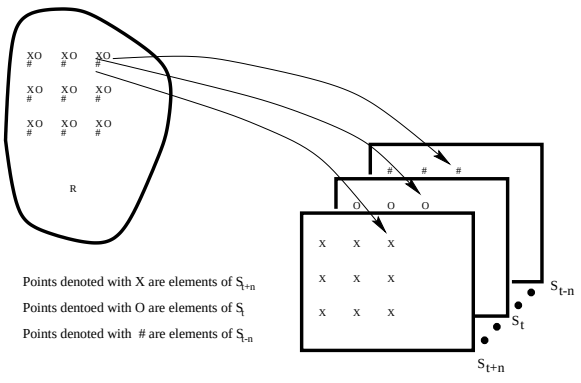

Figure 1. The image sequence that records the region R, registers similar but not identical sets of points in object space (i.e. the original scene).

1.2. Problem Specification

Given a video sequence, we wish to efficiently enhance the resolution of the representation grid in approximated image space, such that the result is the equivalent of having mapped a larger set of points, from the object space to the image space, than actually occurred in any given frame. Further, we would like the resulting set's size to be a (preferably linear) function of the computation expended.

1.3. Approach

We note that by approaching the problem using only spatial context, we can only obtain guesses of the the intensity values that should be assigned to the newly defined points (that result from the higher resolution). Numerous methods exist to do this such as *linear interpolation*, *natural spline interpolation*, or the application of a *Gaussian filter* to a region of points around the one whose intensity is being estimated, for each of the new points. However, sub-pixel features in the original scene may not be recovered by this process.

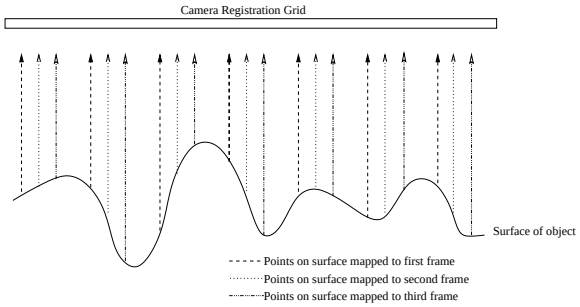

Figure 2. We assume that the camera samples a different set of points on the object in each frame.

Therefore, we use the insight that in consecutive frames, $f_t, \ldots, f_{t+n}$, of the video sequence, the same region R in the object space is captured during the recording process, by mapping different sets of points, $S_t, \ldots, S_{t+n}$, of the object space onto the image space, as illustrated in Figure 2. Thus the sets $S_t, \ldots, S_{t+n}$, are very close approximations of each other, but not identical. This is because the nature of analogue to digital conversion is that of a *many to one* mapping of an *uncountable set* onto a *countable set*.

We will proceed by retrieving information about the region in the object space R, corresponding to the image space I_t under consideration, from representations of the same region R in previous and future frames, $I_{t-n}, \ldots, I_{t-1}, I_{t+1}, \ldots, I_{t+n}$. With a larger set of points mapped from the object space to the image space, we will be in a position to create a higher resolution video frame that captures sub-pixel features of the original object.

1.4. Overview of Algorithm

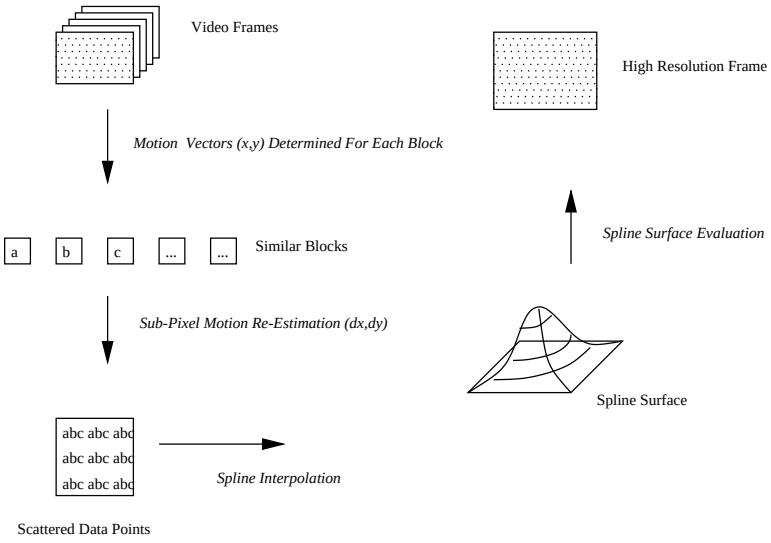

Figure 3. Overview of algorithm.

Each frame of the video sequence is enhanced independently in serial order. To process a given frame, a set of frames is formed by choosing several frames that are immediately previous and several that are immediately after the frame under consideration.

The frame that is being improved is then broken up into rectangular blocks. Extra data about each block in the frame is then extracted. For a given block, this is done by performing a search for one block from each of the frames in the above defined set that best matches it. A sub-pixel displacement that will minimize the error between each of the contextual blocks and the block under consideration is analytically calculated for each of the blocks in the frame.

The data thus obtained can be viewed as a set of points of the form (x, y, z), where z is the image intensity at a point (x, y) in

the XY plane. The set thus obtained will correspond to points scattered in the XY plane. To map this back to a grid, a bivariate spline surface, composed of tensor product basis splines, that approximates this data is computed and evaluated. Since this is done for each frame in the sequence, a sequence of similar length to the input is produced as output.

2. EXTRACTION OF TEMPORAL CONTEXT

Our algorithm enhances the resolution of the video sequence by increasing the size of the data set associated with each frame. Considering one *reference frame*, whose resolution we are currently improving, without loss of generality we will consider the problem of extracting information of relevance from only one other frame, the *contextual frame*. The same process may be applied to a set of other frames immediately before and after the reference frame in the video sequence in order to obtain the increased size data set representing the region in object space that was mapped to the reference frame.

Ideally, whenever an object in the reference frame appears in the contextual frame, we would like to be able to identify the exact set of pixels that correspond to the object, extract the set and incorporate it into the representation of the object in the reference frame. To effect this search, we use a technique from video compression, called *motion estimation*.

2.1. Initial Motion Estimation

Video compression achieves higher compression ratios than still image *intraframe* coding, the process of reducing spatial redundancy in a frame, by hybridization with *interframe* coding, which exploits temporal correlation in the data. This is done by compensating for the motion of objects, from frame to frame, by identifying an object, estimating its motion, and then encoding the movement as a vector representing an *affine transformation*. Currently research on computing optical flow that would yield such representations exists, but for computationally efficient solutions[1] that give a good approximation, standards such as MPEG[2] and H.261 have adopted a different approach.

In the general case, we would allow for a range of motion by the object, taking into account scaling, rotation, and warping in addition to translation. The implication of this is that the geometry of the domain and range of the motion mapping need not be the same. This leads to complex geometrical concerns as well as intensive computational requirements. Standards such as MPEG deal with this by arbitrarily decomposing a frame into blocks and working at the granularity of blocks[3] with the assumption that there is no rotation occurring. This is the equivalent of imposing a uniform motion vector field on all the pixels in the block. To ensure the correctness of the algorithm, a means of encoding the motion of the pixels that do not have the same motion as the block is introduced in the form of a *residual code*, which is the difference of the actual representation and that imposed on the pixel by the block granularity motion vector.

The reference frame is broken into blocks of fixed size. (The MPEG standard specifies the use of 16×16 size blocks.[4] We leave the block dimensions as a parameter of the algorithm to determine the best dimensions for our purpose.) The size of the block must be determined as a function of several factors that necessitate a tradeoff:

- By using smaller blocks, the finer granularity allows for local movements to be better represented. The uniform motion field is forced on a smaller number of pixels.
- The disadvantage is the increase in computational requirements.
- Depending on the matching criterion, smaller blocks may also result in more incorrect matches, since bigger blocks provide more context increasing the probability of correctly determining an object's motion vector.

For each of these blocks, a search for a block of identical dimensions is performed in the contextual frame. A metric that associates cost inversely with the closeness of the match is used to identify the block with the lowest cost. The vector connecting a point on the reference block to the corresponding point of the best match block found in the contextual frame, is determined as the initial motion estimate for that block.

Matching Criteria

If a given video frame is of width F_w and height F_h, and the sequence is T frames long, then a pixel at position (m, n) in the f^{th} frame, is specified by $VS(m, n, f)$, where $m \in \{1, \ldots, F_w\}$, $n \in \{1, \ldots, F_h\}$, and $f \in \{1, \ldots, T\}$. If a given block is of width B_w and height B_h, then a block in the reference frame with its top left pixel at (x, y) is denoted $RB(x, y, t)$, where $x \in \{1, \ldots, (F_w - B_w + 1)\}$, $y \in \{1, \ldots, (F_h - B_h + 1)\}$, and t is the temporal index. For a fixed reference frame block denoted by $RB(x, y, t)$, the block in the contextual frame that has its top left pixel located at the position $(x + a, y + b)$ is denoted by $CB(a, b, u)$, where $(x + a) \in \{1, \ldots, (F_w - B_w + 1)\}$, $(y + b) \in \{1, \ldots, (F_h - B_h + 1)\}$, and where the contextual frame has temporal index $(t + u)$. The determination that the object, represented in the block with its top left corner at (x, y) at time t, has been translated in exactly u frames by the vector (a, b), is associated with a cost given by the function $\mathcal{C}(RB(x, y, t), CB(a, b, u))$.

We chose to use *Mean Absolute Error (MAE)*, or *Mean Absolute Difference (MAD)* as the cost function on the basis of empirical studies in the literature on MPEG standards[1,5] that have shown it works as effectively as more computationally intensive criteria, where

$$MAD(RB(x, y, t), CB(a, b, u)) = \frac{1}{B_w B_h} \sum_{i=0}^{(B_w-1)} \sum_{j=0}^{(B_h-1)} |VS(x + i, y + j, t) - VS(x + a + i, y + b + j, t + u)| \quad (1)$$

is the specific cost function.

Finding the Motion Vector

Given a reference block $RB(x, y, t)$, we wish to determine the block $CB(a, b, u)$, that is the best match possible in the $(t + u)^{th}$ frame, with the motion vector (a, b) constrained to a fixed rectangular search space centered at (x, y). This is illustrated in Figure 4. If this space's dimensions are specified as the parameters p and q of the algorithm, then $a \in \{(x - p), \ldots, (x + p)\}$, and $b \in \{(y - q), \ldots, (y + q)\}$ must hold. For sequences with little motion these values can be made small for computational efficiency. If the motion characteristics of the sequence are unknown, p and q may be set so that the entire frame is searched.

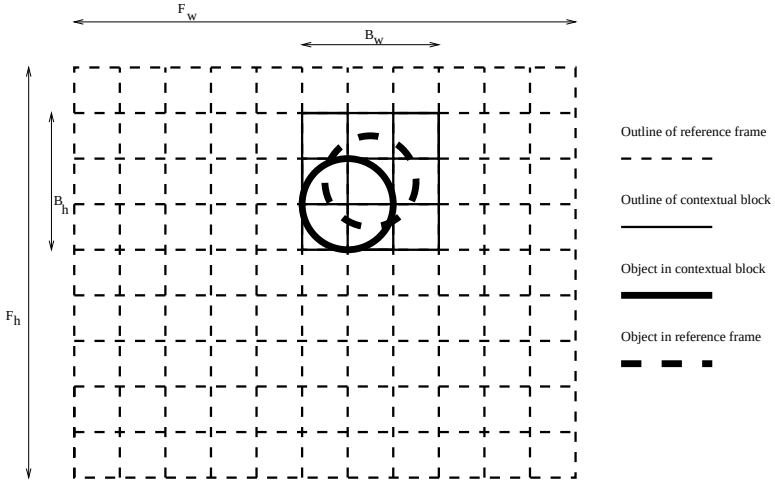

Figure 4. During the first stage, the contextual block containing the object in the reference block is determined with integral pixel accuracy.

The only method that can determine the best choice $CB(a_{min}, b_{min}, u)$ with probability one, such that

$$\begin{aligned} &\forall a \in \{(-x + 1), \ldots, (F_w - B_w - x + 1)\}, \\ &\forall b \in \{(-y + 1), \ldots, (F_h - B_h - y + 1)\}, \\ &\quad \mathcal{C}(RB(x, y, t), CB(a_{min}, b_{min}, u)) \leq \mathcal{C}(RB(x, y, t), CB(a, b, u)) \end{aligned} \quad (2)$$

is *full search*,[6] which calculates $\mathcal{C}(RB(x, y, t), CB(a, b, u))$ for all values of a and b in the above mentioned ranges, and keeps track of the pair (a, b) that has resulted in the lowest value of the cost function. This algorithm has complexity $\mathcal{O}(pqB_wB_h)$ if p and q are specified, and $\mathcal{O}(F_wF_hB_wB_h)$ if the most general form is used.

For general purpose video compression, numerous other algorithms have been developed that can reduce the complexity of the search. The problem with the various alternatives to full search is that they do not guarantee that the correct best match will be returned. They will be useful to us only when there is almost no similarity between the various possible blocks within a given frame, minimizing the chance of incorrect motion estimation.

When video compression is being performed and the motion vector is used in conjunction with a residual code, this is not a problem since the error will be compensated for. However, in our application we are searching

for the specific part of image space that represents the area of object space under consideration. We therefore prefer to use a full search algorithm with limits on p and q if there are computational constraints, since this yields a more effective strategy for our purpose.

In addition, the complexity of context extraction[7] is significantly lower than that of the context integration stage of the algorithm, making it possible to use full search here without noticeably affecting the performance of the implementation.

2.2. Re-estimation With Subpixel Accuracy

Objects do not move in integral pixel displacements. As a result we wish to estimate the motion vector at a finer resolution. Therefore, the initial estimate as outlined above is the first part of a two stage hierarchical search.

We have developed an analytical technique to estimate the motion of a block with sub-pixel accuracy. Its implementation provides the second part. as illustrated in Figure 5. By modeling the cost function, MSE, as a *separable function*, we develop a measure for what real valued displacements along each orthogonal axis will reduce the value of the mean absolute error.

Let the function $\mathcal{F}(x, y)$ contain a representation of the reference frame with temporal index t, that is

$$\forall x \in \{1, \ldots, F_w\}, \forall y \in \{1, \ldots, F_h\} \quad (3)$$
$$\mathcal{F}(x, y) = VS(x, y, t)$$

and let the function $\mathcal{G}(x, y)$ contain a representation of the contextual block for which we are determining the motion vector with sub-pixel accuracy. If $RB(x_0, y_0, t)$ was the reference block for which the previous stage of the algorithm found $CB(a_{min}, b_{min}, u)$ to be the contextual block with the closest match in frame $(t + u)$, then define

$$x_{min} = x_0 + a_{min},\ x_{max} = x_0 + a_{min} + B_w - 1 \quad (4)$$
$$y_{min} = y_0 + b_{min},\ y_{max} = y_0 + b_{min} + B_h - 1$$

and further note that the appropriate definition of $\mathcal{G}(x, y)$ that is required is given by

$$\forall x \in \{x_{min}, \ldots, x_{max}\}, \forall y \in \{y_{min}, \ldots, y_{max}\} \quad (5)$$
$$\mathcal{G}(x, y) = VS(x, y, u)$$

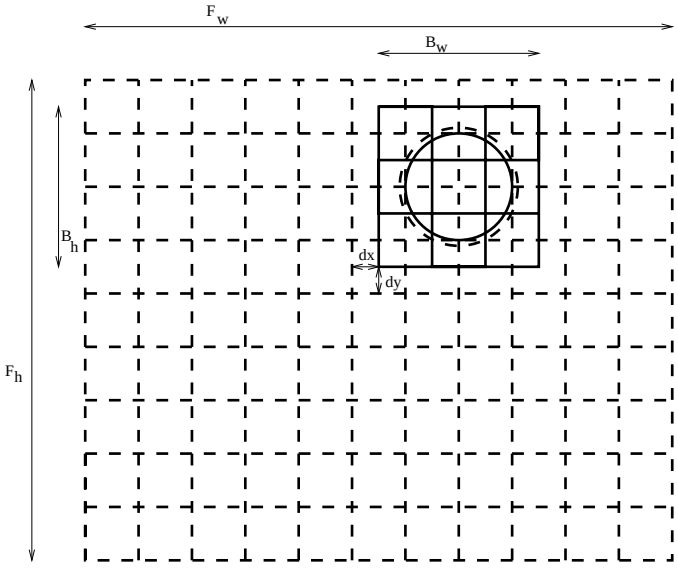

Figure 5. The second stage determines a sub-pixel displacement of the contextual block such that there is an improved match with the reference area.

Instead of (a_{min}, b_{min}), which is the estimate that results from the first stage of the search, we seek to determine the actual motion of the object, which can be represented by $(a_{min} + \delta a_{min}, b_{min} + \delta b_{min})$. Therefore, at this stage the vector $(\delta a_{min}, \delta b_{min})$, which we call the *sub-pixel motion displacement*, must be found.

LEMMA 2.1. *δx can be calculated in $\mathcal{O}(\log(Y B_w B_h))$ space and $\mathcal{O}(B_w B_h)$ time.*

Proof. This statement holds true upto the approximation that intensity values between two adjacent pixels vary linearly. Above, Y is the range of possible values of the function (or, equivalently the range of possible values that a pixel's color can assume).

If we estimate the sub-pixel motion displacement as $(\delta x, \delta y)$, then the cost function is $\mathcal{C}(RB(x, y, t), CB(a + \delta x, b + \delta y, u))$. Thus if we use mean square error as our matching criterion, and we treat it as an independent function of δx and δy, we obtain the equation

$$MSE(\delta x) = \sum_{x=x_{min}}^{x_{max}} \sum_{y=y_{min}}^{y_{max}} [\mathcal{F}(x, y) - \mathcal{G}(x + \delta x, y)]^2 \quad (6)$$

For the purpose of calculating $\mathcal{F}$ and $\mathcal{G}$ with non-integral parameters, we will use linear interpolation. In order to perform such an estimation, the function must be known at the closest integral valued parameters. Since we have values of the pixels adjacent to the reference block easily available, and since this is not true for the contextual block, we reformulate the above equation without loss of generality as

$$MSE(\delta x) = \sum_{x=x_{min}}^{x_{max}} \sum_{y=y_{min}}^{y_{max}} \left[\mathcal{F}(x+\delta x, y) - \mathcal{G}(x,y)\right]^2 \tag{7}$$

By expanding Equation 7 we obtain

$$MSE(\delta x) = \sum_{x=x_{min}}^{x_{max}} \sum_{y=y_{min}}^{y_{max}} \left[\mathcal{F}^2(x+\delta x, y) + \mathcal{G}^2(x,y) - 2\mathcal{F}(x+\delta x, y)\mathcal{G}(x,y)\right] \tag{8}$$

This is the equivalent of using a fixed grid (the contextual block) and variably displacing the entire reference frame, in order to find the best sub-pixel displacement, (instead of the intuitive fixing of the image frame and movement of the block whose motion we are determining).

Assuming that the optimal sub-pixel displacement along the x axis of the contextual block under consideration is the δx that minimizes the function $MSE(\delta x)$ in Equation 8 which models $\mathcal{C}(RB(x_1, y_1, t), CB(a+\delta x, b))$, we proceed to calculate

$$\frac{d}{d(\delta x)} MSE(\delta x) = \sum_{x=x_{min}}^{x_{max}} \sum_{y=y_{min}}^{y_{max}} \left[2\mathcal{F}(x+\delta x, y)\frac{d}{d(\delta x)}\mathcal{F}(x+\delta x, y) - 2\mathcal{G}(x,y)\frac{d}{d(\delta x)}\mathcal{F}(x+\delta x, y)\right] \tag{9}$$

Since we wish to determine the minimum of $MSE(\delta x)$, we will solve for δx using the constraint

$$\frac{d}{d(\delta x)} MSE(\delta x) = 0 = \sum_{x=x_{min}}^{x_{max}} \sum_{y=y_{min}}^{y_{max}} \left[\left[\mathcal{F}(x+\delta x, y) - \mathcal{G}(x,y)\right] \frac{d}{d(\delta x)}\mathcal{F}(x+\delta x, y)\right] \tag{10}$$

Since $\mathcal{F}(x,y)$ is a discrete function, we use linear interpolation to approximate it as a continuous function for the purpose of representing $\mathcal{F}(x+\delta x, y)$ and computing $\frac{d}{d(\delta x)}\mathcal{F}(x+\delta x, y)$.

We consider the case of a positive δx first.

$$\forall \delta x,\ 0 \le \delta x \le 1,$$

$$\mathcal{F}(x+\delta x, y) = \mathcal{F}(x,y) \ + \ \delta x \left[\mathcal{F}(x+1, y) - \mathcal{F}(x,y)\right] \tag{11}$$

$$\frac{d}{d(\delta x)}\mathcal{F}(x+\delta x, y) = \mathcal{F}(x+1, y) - \mathcal{F}(x,y) \tag{12}$$

Applying Equations 11 and 12 to Equation 10, we obtain

$$\{ \sum_{x=x_{min}}^{x_{max}} \sum_{y=y_{min}}^{y_{max}} \left[\left[\mathcal{F}(x+1, y) \ - \ \mathcal{F}(x,y)\right] \left[\mathcal{F}(x,y) \ - \ \mathcal{G}(x,y) \ + \ \delta x \left[\mathcal{F}(x+1, y) \ - \ \mathcal{F}(x,y)\right] \right] \right] \} \ = \ 0 \tag{13}$$

Further expanding, and grouping to separate terms with δx,

$$\{ \sum_{x=x_{min}}^{x_{max}} \sum_{y=y_{min}}^{y_{max}} \left[\delta x \left[\mathcal{F}(x+1, y) - \mathcal{F}(x,y)\right]^2 + \mathcal{F}(x+1, y)\mathcal{F}(x,y) \ - \ \mathcal{F}^2(x,y) + \mathcal{F}(x,y)\mathcal{G}(x,y) \ - \ \mathcal{F}(x+1, y)\mathcal{G}(x,y) \right] \} \ = \ 0 \tag{14}$$

Therefore, by rearranging terms and rewriting the above, we can obtain the closed form solution

$$\delta x = \frac{\sum_{x=x_{min}}^{x_{max}} \sum_{y=y_{min}}^{y_{max}} \left[\left[\mathcal{F}(x+1,y) - \mathcal{F}(x,y)\right] \left[\mathcal{F}(x,y) - \mathcal{G}(x,y)\right] \right]}{\sum_{x=x_{min}}^{x_{max}} \sum_{y=y_{min}}^{y_{max}} \left[\mathcal{F}(x+1,y) - \mathcal{F}(x,y)\right]^2} \tag{15}$$

Similarly an independent calculation can be performed to attempt to ascertain what negative value of δx is optimal. The solution in the the case of a negative δx is a minor variation of the above and is outlined below

$$\forall \delta x,\ -1 \leq \delta x \leq 0,$$

$$\mathcal{F}(x+\delta x, y) = \mathcal{F}(x,y)\ +\ \delta x \left[\mathcal{F}(x,y) - \mathcal{F}(x-1,y)\right] \tag{16}$$

$$\frac{d}{d(\delta x)}(\mathcal{F}(x+\delta x, y)) = \mathcal{F}(x,y) - \mathcal{F}(x-1,y) \tag{17}$$

Apply Equations 16 and 17 to Equation 10 to obtain

$$\{ \sum_{x=x_{min}}^{x_{max}} \sum_{y=y_{min}}^{y_{max}} \left[\left[\mathcal{F}(x,y)\ -\ \mathcal{F}(x-1,y)\right] \right. \tag{18}$$
$$\left. \left[\, \mathcal{F}(x,y)\ -\ \mathcal{G}(x,y)\ +\ \delta x \left[\mathcal{F}(x,y)\ -\ \mathcal{F}(x-1,y)\right] \right] \right\}\ =\ 0$$

After algebraic manipulation, we obtain $\delta x =$

$$\frac{\sum_{x=x_{min}}^{x_{max}} \sum_{y=y_{min}}^{y_{max}} \left[\left[\mathcal{F}(x,y) - \mathcal{F}(x-1,y)\right] \left[\mathcal{F}(x,y) - \mathcal{G}(x,y)\right] \right]}{\sum_{x=x_{min}}^{x_{max}} \sum_{y=y_{min}}^{y_{max}} \left[\mathcal{F}(x,y) - \mathcal{F}(x-1,y)\right]^2} \tag{19}$$

Finally, we compute the MSE between the reference and contextual block, with each of the two δx's. The δx that results in the lower MSE is determined to be the correct one.

The closed form solution for each δx adds $B_w B_h$ terms in both the numerator and the denominator. Each term requires two subtractions and one multiplication. This is followed by computing the final quotient of the numerator and denominator summations. The time complexity is therefore $\mathcal{O}(B_w B_h)$. We note that computing the MSE is possible within this bound as well. Since the function values range upto Y, and only the running sum of the numerator and denominator need to be stored, the space needed is $\mathcal{O}(\log Y B_w B_h)$. □

Next, we note that it is possible to obtain a better representation of a block in the reference frame by completing the analysis of sub-pixel displacements along the orthogonal axis.

LEMMA 2.2. *The cardinality of the set of points that represents the block under consideration from the reference frame is non-decreasing.*

Proof. After a δx has been determined, the block must be translated by a quantity δy along the orthogonal axis. It is important to perform this calculation after the δx translation has been applied, since it guarantees that the MSE after both translations is no more than the MSE after the first translation, ensuring the correctness of the algorithm. We can determine the δy in a manner analogous to the one used to determined δx, using the representation of the MSE below, with the definition $x' = x + \delta x$:

$$MSE(\delta y) = \sum_{x=x_{min}}^{x_{max}} \sum_{y=y_{min}}^{y_{max}} \left[\mathcal{F}(x',y) - \mathcal{G}(x', y+\delta y)\right]^2 \tag{20}$$

This is in turn equivalent to (by the same argument provided for the δx case):

$$MSE(\delta y) = \sum_{x=x_{min}}^{x_{max}} \sum_{y=y_{min}}^{y_{max}} \left[\mathcal{F}(x', y+\delta y) - \mathcal{G}(x',y)\right]^2 \tag{21}$$

Solving for δy is achieved using:

$$\forall \delta y,\ 0 \leq \delta y \leq 1,\ \delta y = \tag{22}$$

$$\forall \delta y,\ -1 \le \delta y \le 0,\ \delta y = \frac{\sum_{x=x_{min}}^{x_{max}} \sum_{y=y_{min}}^{y_{max}} \left[\left[\mathcal{F}(x', y+1) - \mathcal{F}(x', y) \right] \left[\mathcal{F}(x', y) - \mathcal{G}(x', y) \right] \right]}{\sum_{x=x_{min}}^{x_{max}} \sum_{y=y_{min}}^{y_{max}} \left[\mathcal{F}(x', y+1) - \mathcal{F}(x', y) \right]^2} \quad (23)$$

$$\frac{\sum_{x=x_{min}}^{x_{max}} \sum_{y=y_{min}}^{y_{max}} \left[\left[\mathcal{F}(x', y) - \mathcal{F}(x', y-1) \right] \left[\mathcal{F}(x', y) - \mathcal{G}(x', y) \right] \right]}{\sum_{x=x_{min}}^{x_{max}} \sum_{y=y_{min}}^{y_{max}} \left[\mathcal{F}(x', y) - \mathcal{F}(x', y-1) \right]^2}$$

If the sub-pixel displacement $(\delta x, \delta y)$ determined results in an MSE between the contextual and reference blocks exceeding a given threshold, then the extra information is not incorporated into the current set representing the reference block. This prevents the contamination of the set with spurious information. It also completes the proof that either a set of points that enhances the current set is added, or none are - since this yields a non-decreasing cardinality for the set representing the block that is being processed. □

By performing this analysis independently for each of the contextual blocks that corresponds to each of the reference blocks, we obtain a scattered data set representing the frame whose resolution we are enhancing. If the sampling grid had infinite resolution, and we inspected the values registered on it at the points we have determined above to be in the image space, we would find that these data points are a good approximation. By repeating this process with a number of contextual frames for each of the reference frames, we can extract a large set of extra data points in the image space of the reference frame. At this stage, we have analytically enhanced the resolution of the video frame. However, the format of scattered data is not an acceptable format for most display media such as video monitors. Therefore we must process this information further to make it usable.

3. CREATING A COHERENT FRAME

3.1. Framework

We shall define a *uniform grid* to be the set $H_{P,k} \cup V_{Q,l}$ of lines in $\mathbf{R}^2$, where $H_{P,k} = \{x = k\alpha \mid k \in \{0, 1, 2, \ldots, P\},\ \alpha \in \mathbf{R},\ P \in \mathbf{N}\}$ specifies a set of vertical lines and $V_{Q,l} = \{y = l\beta \mid l \in \{0, 1, 2, \ldots, Q\},\ \beta \in \mathbf{R},\ Q \in \mathbf{N}\}$ specifies a set of horizontal lines. Specifying the values of P, Q, α, and β determines the uniform grid uniquely. Given a set S of points of the form (x, y, z), where z represents the intensity of the point (x, y) in image space, if there exist M, N, α, and β such that all the (x, y) of the points in the set S lie on the associated uniform grid, and if every data point (x, y) on the uniform grid has an intensity (that is a z component) associated with it, then we call the set S a *coherent frame*. Each of the frames in the original video sequence was a coherent frame. We seek to create a coherent frame from the data set D, that we have obtained through the process described in Section 2, with the constraint that the number of points in the coherent frame should be the same as that of the data set D.

The general principle we use to effect the transformation from scattered data to a coherent frame is to first construct a surface in $\mathbf{R}^3$ that passes through the input data. By representing this surface in a functional form with the x and y coordinates as parameters, it can then be evaluated at uniformly spaced points in the XY plane for the production of a coherent frame.

There are numerous methods available for the purpose, the tradeoff being the increased computational complexity needed to guarantee a greater level of accuracy of the mapping. While the efficiency of the procedure is important, in the light of our concerns in Section 1.3, we would like to make our approximation as close a fit as possible. We note that although we will be forced to use interpolation techniques at this stage, we are doing so with a data set that has been increased in size as described in Section 2, so this is not equivalent to performing interpolation at the outset and is certainly an improvement over that. We use B-spline interpolation with degree k which can be set as a parameter to our algorithm.

3.2. Implementation Issues

Given an arbitrary scattered data set, we can construct a coherent frame that provides a very good approximation of the surface specified by the original data set. However, if we were to work with the entire data set at hand, our algorithm would not be scalable. This is due to the fact that memory requirement would exceed that which is available on today's computer systems.

Noting that splines use only a fixed number of neighboring points, we employ the technique of decomposing the data set into spatially related sets of fixed size. Each set contains all the points within a block in image

space. The disadvantage of working with such subsets is that visible artifacts develop at the boundaries in the image space of these blocks. To avoid this we *compose* the blocks by using data from adjacent blocks to create a border of data points around the block in question, so that the spline surface constructed for a block is continuous with the surfaces of the adjacent blocks. Working with a block at a time in this manner, we construct surfaces for each region in the image space, and evaluate the surface on a uniform grid, to obtain the representation we desire. When this is done for the entire set, we have obtained a coherent frame.

LEMMA 3.1. *Creating a block in a coherent frame requires $\mathcal{O}(B_w B_h \sigma(k) T)$ operations using data extracted from temporal context.*

Proof. Here T frames are incorporated, and degree k polynomial based B-spline tensor products are used to perform the transformation from scattered to gridded data. Lemma 2.2 guarantees that the input set does not degenerate. The complexity of splining depends on the number of points, which is $B_w B_h$, the product of the block width and height. $\sigma(k)$ is the cost to perform the splining operations per point in the data set.

Since B-splines have the property of *local support*, that is only a fixed number of adjacent B-splines are required for the evaluation of any given point in the space spanned by them (such as the surface being represented), and each B-spline can be represented as a fixed length vector of coefficients, the approximation of a surface specified by a set of points has time complexity that is only bound by the degree of the polynomials used and the multiplicity of the knots.[8,9] The above result follows immediately from this. □

4. RESULTS

4.1. Time Complexity

THEOREM 4.1. *The resolution of a video sequence can be enhanced in $O(n)$ time, where n is the size (in bits) of the raw video data, if:*
(a) the degree of splines used in interpolation is fixed,
(b) a constant number of frames of temporal context is used, and
(c) the range of motion estimation is limited to a fixed multiple of the block size.

Proof. We assume that there are L frames to process, in each of which there are $\frac{F_w F_h}{B_w B_h}$ blocks. For each block, it takes $\mu(B_w, B_h)$ time to perform motion estimation (assuming fixed range exhaustive search) and re-estimation of the motion with sub-pixel accuracy, by Lemma 2.1. Transforming the data set obtained for a block takes $\mathcal{O}(B_w B_h \sigma(k) T)$ time, by Lemma 3.1.

Therefore, processing an L frame video sequence using T frames of temporal context to enhance the resolution of each frame, yields the higher resolution version in $\mathcal{O}(F_w F_h L[\frac{\mu(B_w, B_h)}{B_w B_h} + \sigma(k) T])$ time.

By limiting the range of motion estimation to a fixed multiple of the block size, $\mu(B_w, B_h) = O(1)$. Using a constant number of frames of temporal context results in $T = O(1)$. Finally, while in theory B-spline interpolation has complexity $\mathcal{O}(k \log^2 k)$, constructing a B-spline as well as evaluating it along with all its derivatives can be done in $\mathcal{O}(k^2)$ operations in practice. However, if k is fixed, then $\sigma(k) = O(1)$.

Since $n = F_w F_h L$ and if the above specified constraints hold, then enhancing the resolution of the video sequence has a time complexity bound of $O(n)$. □

4.2. Application GROW

The program GROW was developed as an implementation of the algorithm. It provides a flexible command line interface that allows the individual specification of numerous parameters such as the horizontal and vertical dimensions of the blocks used for motion estimation, the blocks used for spline interpolation, the extra border used for composing blocks for spline interpolation, the degrees of the splines used, the factors by which the output frame is scaled up from the original, the maximum range in which to perform motion estimation, the number of previous and later frames used to enhance any given frame, the error threshold that is acceptable during motion estimation and that for spline interpolation.

The parameters entered serve to guide but not enforce the algorithm. For example, splining starts with the degree the user enters but automatically drops to lower degrees (as less context is used) when the surface returned is not close enough to the points it is approximating. The error threshold specified for splining is used to scale up bounds that are calculated using heuristics from the literature.[10] Intermediate results, such as the sub-pixel displacements, are calculated and kept in high precision floating point format. When error thresholds

Figure 6. Original frames.

are crossed, the data in question is not used. Thus occlusion and peripheral loss of objects is dealt with by the effective result of using only reference image data for the relevant region.

4.3. Experiments

The methodology employed to test the program is specified below. A video sequence was sub-sampled, on a frame by frame basis. The new lower resolution sequence was used as data, and the output compared to the original sequence.

If $\mathcal{G}$ represents the high resolution video sequence, and $\mathcal{F}$ represents the high resolution sequence obtained by running GROW on the low resolution version associated with $\mathcal{G}$, where the low resolution version was obtained through the sub-sampling of $\mathcal{G}$, then the *Signal to Noise Ratio* is defined as:

$$SNR = 10 \log \frac{\sqrt{\frac{1}{B_w B_h} \sum_{x=x_{min}}^{x_{max}} \sum_{y=y_{min}}^{y_{max}} \mathcal{F}(x,y)}}{\frac{1}{(B_w B_h)^2} \sum_{x=x_{min}}^{x_{max}} \sum_{y=y_{min}}^{y_{max}} [\mathcal{F}(x,y) - \mathcal{G}(x,y)]^2} \tag{24}$$

Using this metric to measure distortion, we compare the strength of the signal of the sequence produced by GROW under various cicrumstances.

As an example, note the parts of five smaller frames in Figure 4.2 that are part of the original video sequence, **garden**. By using all of them as input to GROW, the higher resolution output corresponding to the the third original frame is generated and shown. Figure 4.3 shows versions constructed with GROW as well as with spline interpolation, along with the original frame. Artifacts that arise in the final images are due to errors in initial block matching and splining.

We note that the improvement over spline interpolation by the addition of data obtained from temporal context is done at little practical cost. This is demonstrated in the data in Figure 7 by the fact that the amount of time spent on this is minimal compared to time spent on the spline construction and evaluation.

The effect of varying the parameters of the algorithm is an observable as well as measurable change in the quality of the output images, as measured by the SNR heuristic.

In Table 1, the first line represent the use of only spline interpolation without any proximal frames. The second line uses 2 previous and 2 later frames. The SNR is noticeably higher. The third line also uses a total of 5 frames, but the

Figure 7. Time spent broken down by splining and temporal context extraction.

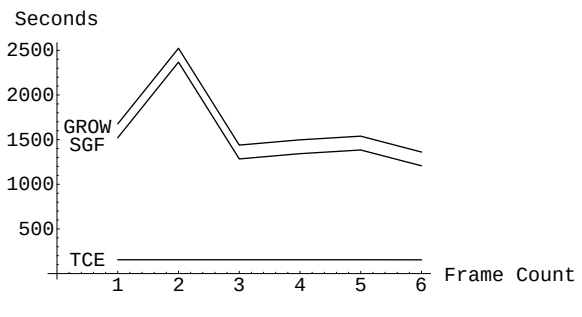

Figure 8. The frame whose resolution is being increased.

Figure 9. The same frame with its resolution increased using only spline interpolation.

Figure 10. Higher resolution version of the central frame obtained using GROW.

motion estimation blocks are 4x4 instead of 8x8 as in the second case or 16x16 in the first case. This creates a further improvement in the SNR (at the cost of increased computation). Only 4 parameters are shown - the range for initial motion estimation on each axis and the number of earlier and later frames used, respectively. However, GROW includes 13 other parameters that can be adjusted. We describe them in Appendix A.

s	t	j	k	x	y	c	d	p	l	m	n	a	b	z	u	e	SNR
16	16	4	4	8	8	2	2	0	0	64	64	5	5	1	0	4	28.39
8	8	4	4	8	8	2	2	2	2	64	64	5	5	1	0	4	30.40
4	4	4	4	8	8	2	2	2	2	64	64	5	5	1	0	4	30.59

Table 1. Effect of varying the parameters on the output frame's SNR

As mentioned in the overview, certain aspects such as spline degree selection have been automated, while others such as splining error bounds are semi-automatically calculated, using heuristics and minimal user input. To get an optimal image sequence, though, it is necessary for the user to manually adjust the values fed into the algorithm. Finally, we note that by hand tuning GROW, we can obtain a significantly better result than possible by spline interpolation without using temporal context. This can be noted from the table as the row which uses no past and future frames effects this kind of spline interpolation, but does not yield as strong a signal as the best case output of GROW.

5. RELATION TO PREVIOUS WORK

Identifying points in object space using image sequences has been dealt with extensively in the literature on computer vision, geometric modeling, image processing, and pattern recognition. Work has even been done on tracking objects with a moving camera.[11] To deal with the problems of discerning rotation from translation when the motion is small, as well the sensitivity of the measurements of depth (to determine object shape) to noise, factorization methods that analytically separate the motion from the object have been developed for the cases where the motion is planar,[12] or in $\mathbf{R}^3$,[13] as well as when the projection is either orthographic,[14,15] under perspective[16] or the paraperspective case.[17] The analyses have been carried out within a framework where either there exists a fixed model of camera or object motion, or the study seeks to identify the parameters[18] that constitute the mapping $\mathcal{M}$, from object space to image space.

Applying such methods would involve identifying $\mathcal{M}$ explicitly and computing $\mathcal{M}^{-1}$, or at the very least determining $\mathcal{M}^{-1}$ empirically. This would be needed in order to capture representations in object space, that could then be transformed to image space. By increasing the information available, a higher resolution frame could then be constructed. However, we seek to develop an *efficient* algorithm for video resolution enhancement,

by which we mean that we wish to obviate the need for explicit mappings back to object space. Our algorithm will achieve the extraction of extra data points in any given frame by working directly with the frame's image space, and the associated *temporal context*, the past and future frames.

Digital signal processing techniques target the problem of enhancing the resolution of the video sequence.[19] In contrast to other approaches, our work is explicitly parametrized so that cost and quality can be traded as needed. Further, we undertake an explicit analysis of the time complexity of the reconstruction process. Our approach does not account for factors such as camera blur, occlusion or the aperture problem.[20]

6. CONCLUSION

This paper describes how temporally proximal frames of a video sequence can be utilized to aid the forensic analyst by enhancing the resolution of individual frames. The framework developed allows the analyst to vary a number of parameters so that a tradeoff may be made between the quality of the reconstructed frame and the time it takes to effect the process.

The algorithm's time complexity is analyzed as it is being developed. Additionally, we describe our implementation of it as a C program that takes a video sequence as input and produces a higher resolution version as the output. Finally, we describe an example of its application, where varying the parameters results in an increased signal-to-noise ratio in the output as compared to the input.

REFERENCES

1. V. Bhaskaran and K. Konstantinides, *Image and Video Compression Standards: Algorithms and Architectures*, Kulwer Academic Publishers, Boston, MA, 1996.
2. K. Challapali, X. Lebegue, J. Lim, W. Paik, R. S. Girons, E. Petajan, V. Sathe, P. Snopko, and J. Zdepski, "The grand alliance for us hdtv," *Proceedings of the IEEE* , 1995.
3. D. L. Gall, "Mpeg: A video compression standard for multimedia applications," *Communications of the ACM* , 1991.
4. L. Chiariglione, "The development of an integrated audiovisual coding standard: Mpeg," *Proceedings of the IEEE* , 1995.
5. T. Markas, *Data Compression: Algorithms and Architectures*. PhD thesis, Duke University, 1993.
6. K.-W. Wang, R.-L. Wang, and Y.-X. Yan, "Optoelectronic hybrid system for the full-search block-matching motion compensation algorithm."
7. P. Pirsch, N. Demassieux, and W. Gehrke, "Vlsi architectures for video compression - a survey," *Proceedings of the IEEE* , 1995.
8. C. de Boor, *A Practical Guide to Splines*, Springer-Verlag, New York NY, 1978.
9. L. Schumaker, *Spline Functions Basic Theory*, John Wiley and Sons, New York, NY, 1981.
10. P. Dierckx, *Curve and Surface Fitting with Splines*, Oxford University Press, New York, NY, 1993.
11. Burt, Bergen, Hingorani, Kolczynski, Lee, Leung, Lubin, and Shvayster, "Object tracking with a moving camera - an application of dynamic motion analysis," *Proceedings of the IEEE* , 1989.
12. C. Tomasi and T. Kanade, "Shape and motion from image streams: A factorization method - planar motion," tech. rep., Carnegie Mellon University, 1990.
13. C. Tomasi and T. Kanade, "Shape and motion from image streams: A factorization method - point features in 3d motion," tech. rep., Carnegie Mellon University, 1991.
14. C. Tomasi and T. Kanade, "Shape and motion from image streams: A factorization method - full report on the orthographic case," tech. rep., Carnegie Mellon University, 1992.
15. C. Tomasi and T. Kanade, "Shape and motion from image streams under orthography: A factorization method," *International Journal of Computer Vision* , 1992.
16. C. Tomasi, "Pictures and trails: A new framework for the computation of shape and motion from perspecitve image sequences," tech. rep., Cornell University, 1993.
17. C. Poelman and T. Kanade, "A paraperspective factorization method for shape and motion recovery."
18. T.-H. Wu and R. Chellapa, "Experiments on estimating motion and structure parameters using long monocular image sequences," tech. rep., University of Maryland, College Park, 1992.
19. N. R. Shah and A. Zakhor, "Resolution enhancement of color video sequences," *IEEE Transactions on Image Processing* **8(6)**, pp. 879–885, 1999.
20. S. Borman, *Topics in Multi-frame Super-resolution Restoration*. PhD thesis, University of Notre Dame, 2004.

APPENDIX A. *GROW* PARAMETERS

```
Usage:  grow
        -f <first-frame> -g <last-frame> -h <frame-height>
        -w <frame-width> -i <input-files> -o <output-files>
        [-s <x-dimension-of-splining-blocks>]
        [-t <y-dim-spl-blks>]
        [-j <x-dim-extra-pixels-for-splining>]
        [-k <y-dim-extra-pixels>]
        [-x <x-dim-of-motion-estimation-blocks>]
        [-y <y-dim-mot-est-blks>]
        [-c <horizontal-scaling-factor>]
        [-d <vertical-scaling-factor>]
        [-p <number-of-previous-frames-used-as-context>]
        [-l <num-later-frms>]
        [-m <maximum-x-displacement-during-motion-est>]
        [-n <max-y-disp>]
        [-a <x-dimension-spline-degree>]
        [-b <y-dimension-spline-degree>]
        [-z <spline-acceptable-error-factor>]
        [-u <class-of-max-accept-err>]
        [-e <maximum-error-during-motion-est>]
        [-v (verbose)]
```

-f The index number of the first frame where processing of the video sequence is to begin. For example, if the entire sequence of files $frame1.Y, \ldots, frame20.Y$ is to be processed then the value used as a parameter of -f would be 1.

-g The index number of the last frame to be processed. In the above example, the parameter of -g would be 20.

-h The integer number of pixels in the vertical direction of each individual frame in the video sequence.

-w The integer number of pixels in the horizontal direction of each individual frame in the video sequence.

-i The string name of the input sequence of files to be used. For example, to process the sequence $frame1.Y, \ldots,$ $frame20.Y$, the parameter $frame$ would be used for the -i option.

-o The string name of the output video sequence. For example, if the parameter $newframe$ was used for the -o option, then if there are 5 output frames, they will be stored as $newframe0.Y, \ldots, newframe4.Y$.

The options above must be specified by the user. The ones below have defaults.

-s The integer parameter supplied is the horizontal dimension of the block size used during spline interpolation and evaluation.

-t The integer parameter is the vertical dimension of the block size used during spline interpolation and evaluation.

-j During spline interpolation of a block, pixels from adjacent blocks are used to sew the blocks together. The integer parameter supplied is the horizontal width of the border that runs parallel to the vertical axis.

-k The integer parameter supplied is the vertical width of the above border that runs parallel to the horizontal axis.

-x The integer parameter supplied is the horizontal dimension of the block size used during motion estimation.

-y The integer parameter is the vertical dimension of the block size used during motion estimation.

-c The integer parameter supplied specifies the factor by which the horizontal resolution is increased when evaluating the spline surface representing the image.

-d The integer parameter supplied specifies the factor by which the vertical resolution is increased when evaluating the spline surface.

-p The integer parameter is the number of previous frames that are used during temporal context extraction to add information to the data set representing the frame that is currently being processed.

-l The integer parameter is the number of later frames that are used to add information for the frame that is currently being processed.

-m The integer parameter supplied is the maximum horizontal direction in which to search for a motion vector. If this is set to 0, then a full search throughout the entire image is performed.

-n The integer parameter supplied is the maximum vertical direction in which to search for a motion vector. If this is set to 0, then a full search throughout the entire image is performed.

-a The integer parameter is the maximum degree of the polynomials used in the horizontal direction. If this yields unacceptable results, the application will automatically try lower degree polynomials as well.

-b The integer parameter is the maximum degree of the polynomials used in the vertical direction. If this yields unacceptable results, the application will automatically try lower degree polynomials as well.

-z During spline interpolation the smoothing norm is subject to a least mean squares constraint. The floating point parameter supplied here specifies that value.

-u The spline routines from SURFIT provide a return value indicating what kind of approximation was performed. The integer parameter supplied here specifies the level of approximation considered acceptable. This should usually be left at 0.

-e During motion estimation, if the MAE between the reference and currently considered blocks is below the integer parameter supplied here, then an acceptable match is considered to be found.

-v When this option is used the application sends detailed information about what it is currently doing to the standard output.