A 3-d Error Diffusion Dither Algorithm for Half-Tone Animation on Bitmap Screens

Hermann Hild and Markus Pins

ABSTRACT

Mapping continuous-tone pictures into digital halftone pictures or color-reduced pictures is a well explored technique. Such algorithms are needed whenever displaying continuous-tone pictures on graphic devices with color levels less than in the original picture. Usual algorithms are designed for single pictures, and they perform poorly when applied to an animated sequence of pictures. They produce correct but different pixel-patterns for each single picture, therefore creating a considerable amount of noise in the moving sequence. This paper examines this phenomenon and proposes a modification of the two-dimensional error diffusion dither algorithm which is able to reduce the described noise while maintaining a high picture quality.

Keywords: dithering, digital halftone picture, animation of dithered pictures

1 DIGITAL HALFTONE ANIMATION

Bitmap displays are widely used for powerful user-friendly interfaces to systems in areas like software engineering, office systems, and CAD. One type of graphical elements used are still pictures (e.g. for icons) originating from a graphics editor, or more advanced, from a synthetic source (e.g. raytracing) or a natural source (e.g. a camera). Usually these pictures are graytones, i.e. every pixel may possess one out of a finite number of different gray levels. This causes the necessity of a reduction to the bitmap quality of the screen. Algorithms mapping graytone pictures into binary pictures are called **dither algorithms**. In a **digital halftone** or **binary picture** every pixel may have only two values, black or white. In the following we will refer to white and black pixels as set or not set pixels, respectively.
Todays workstations are capable to display a sequence of several dozens of bitmapped pictures of a size of, say, 200×200, at a rate of 16 images or more per second. This is sufficient to give the observer the impression of continuous motion, thus opening the user of a bitmap workstation further possibilities. If the movie is initially given in graytone, an immediate way of dithering is to use one of the many existing dither algorithms to map each graytone picture independently into the corresponding binary picture. However, an undesirable side effect may occur in the form of a noisy *flickering*. This is due to the fact that similar areas in successive graytone pictures may be mapped into pixel-patterns in the binary picture which are not similar at all. Despite the underlying suggestion of regularity, in this text we use the word *pattern* to refer to *any* arrangements of pixels. Thus every single binary picture has patterns which correctly represent the corresponding gray values in the graytone pictures, but in the moving sequence these patterns may heavily change over time, causing remarkable noise.

Unfortunately this effect cannot be demonstrated without a bitmap display, nor can one visually figure out changing patterns in subsequent pictures. We denote the **difference-picture** of two binary pictures $B1$ and $B2$ as the picture in which a pixel is set if and only if the values of the pixels at the two corresponding positions in $B1$ and $B2$ differ. Difference-pictures allow some conclusions on the behavior of the pictures: the more set pixels in the difference-picture, the more change and therefore flickering will be in the sequence. Of course changing pixels disturb more or less, depending on their context in the picture. Changing pixels do not at all disturb areas where genuine changes in the graytone pictures take place. Figure 1 shows two successive pictures of a raytraced picture sequence. The only moving parts in the sequence are the wings and their shadows. The background and the tower remain the same patterns over the whole sequence. Nevertheless the patterns in the background are different in each binary picture, as indicated by the difference-picture in figure 2.

Figure 1: Subsequent pictures from the sequence *windmill*

Each of the pictures is correctly dithered. The different pixel-patterns which cause the flickering cannot be recognized by a visual comparison.

The previous example was dithered with one out of many possible algorithms. However, this behavior is shown by all algorithms that represent similar areas in the graytone pictures by different pixel-patterns in the binary pictures. Typical representatives are the *Floyd-Steinberg Algorithm* [FS75], the *Dot Diffusion Algorithm* [Knu87], or the *Constant-Level-Thresholding and Two-dimensional error-diffusion* [Stu82]. For this type of algorithm, a straightforward solution to achieve non-flickering sequences is slightly shifting the pixels set in the actual binary picture in order to achieve pixel-patterns as close as possible to those of the previous picture. As a boundary condition, the number and distances of the pixel movements is to be minimized. Experiments show that even if every pixel is shifted by at most *one* position, more than two third of the pixels can be made to agree. However, the shift by just one position essentially spoils the picture, contours become less defined and regular pixel-patterns become irregular. Further, the movement by only one position is not enough to remedy the flickering problem.
Another class of dither algorithms places the pixels independently of their environment in the graytone or binary picture always creating the same pixel-patterns from the same graytone-areas. If these algorithms additionally show a certain steadyness, i.e. similar gray areas are mapped into similar pixel-patterns, then they indeed create sequences without the disturbing

Figure 2: The resulting difference-picture

It demonstrates that there are no major areas which have the same pixel-patterns.

flickering. The *ordered-dither* algorithms, for example dithering with a *dither matrix* [ES86] [Knu87], satisfy these properties, and they indeed create sequences without flickering. The problem of these algorithms is that they do not generate pictures of a quality as high as those algorithms which are suffering from the flickering problem.
The remainder of this paper describes a modification of the *two-dimensional error diffusion* algorithm reducing the problem of flickering. This algorithm of the first class was chosen due to its flexibility. It seems adaptable to all situations where dithering is necessary. The image quality is better than that of the *dither matrix approach* [Stu82] and as good as the *dot diffusion method* described by [Knu87]. Section 2 introduces into the two-dimensional error diffusion algorithm. In section 3, its adaption to sequences of pictures is described. This adaption can be seen as a 3-d extension of error diffusion. It introduces a sometimes undesired motion blur effect which can be remedied by suggestions presented in section 4.

2 DITHERING WITH CONSTANT-LEVEL THRESHOLDING AND TWO-DIMENSIONAL ERROR-DIFFUSION

Dithering with *Constant-Level Thresholding and Two-dimensional error-diffusion* as described by [Stu82] is an extension of the *Floyd-Steinberg-Algorithm* [FS75]. At every position (i,j) of a picture P, a carried error value resulting from the weighted average of previously computed errors is added. The range of picture values $P(i,j)$ comprises the values $[Graylevel_{min}, Graylevel_{max}]$. The errorcarry is computed by a weighted errorfilter. The weight coefficients are chosen to be 2^n to improve computational efficiency.
A point in the bitmap is inserted if the graylevel value at position (i,j) including $ErrorCarry(i,j)$ is less or equal to threshold $\frac{Graylevel_{max}-Graylevel_{min}}{2}$, i.e.

$$Bitmap(i,j) = \begin{cases} 0 & if\ P(i,j) + ErrorCarry(i,j) > \frac{Graylevel_{max}-Graylevel_{min}}{2} \\ 1 & if\ P(i,j) + ErrorCarry(i,j) \leq \frac{Graylevel_{max}-Graylevel_{min}}{2}. \end{cases}$$

The $Errorcarry$ value results from the weighted average of previously computed $Errors$:

$$ErrorCarry(i,j) = \frac{\sum_{(k,l)\in Area} Error(i+k, j+l) * Weight(k,l)}{\sum_{(k,l)\in Area} Weight(k,l)}$$

The weight function *Weight* and the environment *Area* of summation are defined by

```
             Weight                                Area


    .   .   .   .   .   .   .
    .   1   2   4   2   1   .           { (-2, -2), (-2, -1), ...  (-2, 2),
    .   2   4   8   4   2   .             (-1, -2), (-1, -1), ...  (-1, 2),
    .   4   8 (i,j) .   .   .             ( 0, -2), ( 0, -1) }
    .   .   .   .   .   .   .
```

The values of *ErrorCarry* depend on the function *Error*. *Error* itself depends on whether a point in the *Bitmap* has been inserted or not. At position (i,j), we want to approximate the graylevel value $P(i,j) + ErrorCarry(i,j)$. If a point at this position is inserted in the *Bitmap* and $P(i,j) + ErrorCarry(i,j) > Graylevel_{min}$, the dithered picture is too dark by the amount of $P(i,j) + ErrorCarry(i,j) - Graylevel_{min}$. If the point at position (i,j) is not inserted, the dithered picture is too bright by the amount of $P(i,j) + ErrorCarry(i,j) - Graylevel_{max}$. Hence

$$Error(i,j) = \begin{cases} P(i,j) + ErrorCarry(i,j) - Graylevel_{min} & ,if\ Bitmap(i,j) = 0 \\ P(i,j) + ErrorCarry(i,j) - Graylevel_{max} & ,if\ Bitmap(i,j) = 1. \end{cases}$$

When dithering images with homogenous background it is recommendable to superpose the image with a random noise function to avoid regular patterns. The random noise can be obtained by using a random number generator, creating random numbers in the range from $-0.05 * \frac{Graylevel_{max}-Graylevel_{min}}{2}$ to $+0.05 * \frac{Graylevel_{max}-Graylevel_{min}}{2}$. Generating numbers with a bigger amplitude leads to falsifications of the image. These are added to the image values, i.e.

$$Bitmap(i,j) = \begin{cases} 0 & if\ P(i,j) + ErrorCarry(i,j) + noise > \frac{Graylevel_{max}-Graylevel_{min}}{2} \\ 1 & if\ P(i,j) + ErrorCarry(i,j) + noise \leq \frac{Graylevel_{max}-Graylevel_{min}}{2}. \end{cases}$$

3 DIGITAL HALFTONING OF PICTURE SEQUENCES

Since gray levels in the graytone picture can only be represented by the rough approximation of black or white pixels, an error occurs almost always when a pixel is set in the binary picture. The principle of the error diffusion algorithm is to take such an error into account in the further process of deciding whether a pixel is to be set or not. This is extended to picture sequences by facilitating the decision for a pixel to be set if there is a set pixel in the precedent picture and vice verse. The goal is to achieve the *best matching pixel-patterns* in two subsequent pictures, expressed by the *Hamming-distance* between two pictures.
This can easily be realized. We enlarge the probability for an unset pixel by lowering the **threshold** by a constant *Delta* if there is an unset pixel in the previous binary picture. If the corresponding pixel in the previous picture is set, we raise the **threshold** by adding *Delta*, thus lowering the change that a pixel will not be set. In details,

$$Bitmap(i,j,t) = \begin{cases} 0 & if\ P(i,j,t) + ErrorCarry(i,j,t) + Delta(i,j,t-1) > \frac{Graylevel_{max} - Graylevel_{min}}{2} \\ 1 & if\ P(i,j,t) + ErrorCarry(i,j,t) + Delta(i,j,t-1) \leq \frac{Graylevel_{max} - Graylevel_{min}}{2} \end{cases}$$

with

$$Delta(i,j,t) = \begin{cases} -Delta & if\ Bitmap(i,j,t) = 1 \\ Delta & if\ Bitmap(i,j,t) = 0 \end{cases}$$

Of course we have to check whether this manipulation of the threshold will still produce correct pictures:

- □ Neither constant *raising*, *lowering* or *varying* the threshold will affect the quality of the dithered binary pictures except in very local areas.

  This can be explained by the *feedback* property of the algorithm. An unsymmetric threshold will indeed cause a *wrong* decision at a local position. However, the greater error due to this decision is not lost because it causes an automatic correction by influencing the errorcorrection function.

- □ Setting pixels according to the previous picture rather than to the needs of the actual picture increase the errors made, but even in the worst case the $ErrorCarry$ at any position (i,j) is bound,

  $$(*) \quad |ErrorCarry(i,j)| \leq \frac{Graylevel_{max} - Graylevel_{min}}{2} + Delta.$$

  The function $ErrorCarry(i,j)$ has the same bounds since it is the weighted average of some values $Error(i',j')$ in the local neighborhood of (i,j).

  This can easily be proven by induction:

  - ○ At the first position the error is 0 by definition.

  - ○ For all positions (i,j): if all made errors are bounded by $(*)$, then we can show that the error of the following position is also bounded by $(*)$. This relies on the fact that $ErrorCarry(i,j)$ is bounded by $Error(i,j)$. An evaluation of the possible results for the new error yields the asserted bounds.

Figure 3 demonstrates the result of the application of this algorithm. The value for *Delta* was chosen at 15% of the range of the gray values. The picture shows that already this relatively small *Delta* is able to keep almost all pixels at their previous positions. A visual judgement of the moving sequence shows that the problem of flickering is sufficiently solved.
In the picture of figure 3, structures of previous pictures can be realized in the actual picture. The motion blur effect thus introduced might be useful in case of computer generated graphics minimizing the effect of temporal aliasing. Nevertheless this can be interpreted as a deficiency of the algorithm seen from a single image. In the next section, a modification of the algorithm is presented eliminating this effect.

Figure 3: Picture out of the sequence *windmill*

The picture is dithered with the refinement just described. At the wings one can clearly see the remainders of previous pictures.

4 THE CHANGE-PICTURE TECHNIQUE

The above algorithm can successfully keep most of the pixel-patterns constant, but it is a mistake to try to keep patterns constant if real changes are happening in the graytone pictures. This is demonstrated by the remaining wings in figure 3. The first modification that is suggested now is to apply the pattern-keeping threshold manipulation only in regions where no or only minor changes happened between the previous and the actual graytone picture. A measure for the changes between to subsequent graytone pictures is given by

$$Change(i,j,t) = |P(i,j,t) - P(i,j,t-1)|$$

This yields the following formula for the value of $Bitmap(i,j,t)$,

$$Bitmap(i,j,t) = \begin{cases} 0 & if\ P(i,j,t) + ErrorCarry(i,j,t) + Delta(i,j,t-1) > \frac{Graylevel_{max} - Graylevel_{min}}{2} \\ 1 & if\ P(i,j,t) + ErrorCarry(i,j,t) + Delta(i,j,t-1) \leq \frac{Graylevel_{max} - Graylevel_{min}}{2} \end{cases}$$

Figure 4: Difference-Picture out of the refined sequence

The only differences in the difference-picture are at the moving parts of the picture, the wings and their shadows.

with

$$Delta(i,j,t) = \begin{cases} -w(Change(i,j,t)) * (Graylevel_{max} - Graylevel_{min}) & if\ Bitmap(i,j,t) = 1 \\ w(Change(i,j,t)) * (Graylevel_{max} - Graylevel_{min}) & if\ Bitmap(i,j,t) = 0 \end{cases}$$

The function w determines to what extent the algorithm tries to copy the values of the pixels in the previous picture. The goal is to keep patterns with a maximum extent if no change takes place at a position (i,j) (i.e. $Change(i,j,t) = 0$). With increasing changes, w should become smaller. Some concrete values for w are shown in figure 5.
Unfortunately, this modification is not sufficient to overcome the *structure-keeping* property of the algorithm. The refined algorithm does not try to keep pixel-patterns fixed in areas where the image content of two subsequent graytone pictures changed. Nevertheless it can be still observed that it tends to generate patterns which are similar to structures in previous pictures. This can be explained by a certain *inertia* of the error diffusion algorithm in general. (Spatial) changes in the gray level within a graytone picture may effect a larger area in the binary picture than the exactly corresponding positions in the graytone picture.
We demonstrate this by regarding the lower-left wing of the mill: in figure 1, one can realize that there are no set pixels at all within the facets of the wing. Instead of *white holes* one should expect the same patterns as in the surrounding background, since the background in the graytone picture is the same over the whole area of this wing. These *holes* are the areas where the *wrong* old structures occur: the algorithm tries to keep the pixel-patterns in the facets of the wing, since the gray level of this areas did not change. By doing so, the undesired *holes* are kept. The disturbing results can be observed in figure 3.
The problem was caused by the shadow-like disturbances created by the error diffusion algorithm. It should not be tried to keep pixel-patterns fixed in such areas. Therefore *safety-zone* is now built around all areas changing in time, by expanding the borders of this areas. Ap-

```
        Delta
      +
100%  +
      +
      +

      .
      .

      +
 15%  *
      +
      +
      +
      +
      +
  5%  +             *
      +
      +
      +-------------+-------------+----------  ....  --*---->
     (0             13                                255)
 Graylevel_min                                   Graylevel_max
```

Figure 5: Weighting function w

plying this dilatation to *Change* yields

$$Change'_k(i,j,t) = \max_{i-k \le n \le i+k, j-k \le m \le j+k} (Change(n,m,t)),$$

where k determines the size of the dilatation-filter.
Making the threshold manipulation dependent on $Change'$ instead of *Change* excludes the critical areas around changing structures from the areas where the pixel-patterns are tried to be fixed.
Figure 6 shows a picture out of a sequence dithered with the additional application of the dilatation. The graytone (*CHANGE*)-picture was dilated with a filter of size $k = 7$. Besides some casual disturbances in a pixel-pattern there are no more structures of previous pictures.

APPLICATION

The proposed algorithm showed to be a helpful tool to dither non-flickering animations. The responsibility of the function w is to determine the amount of influence of the previous picture. For the windmill-example we achieved the best results with values for w as presented above. However, depending on the image content other values of w might improve the result.
The memory and execution time requirements are linear in the size of the images. Because the programm is written in *PASCAL*, the execution time can be optimized easily. Approximately 10 seconds on a *SUN 3-50* are required for a 200×200 image.
Since the resulting pixel-patterns are always the result of a compromise between the optimal pixel-pattern for the actual picture and the pattern of the previous picture, some casual irregularities in the patterns are unavoidable. When the algorithm comes from such an area into an area with a given pattern, it suddenly has to adopt to this pattern. These are critical areas. To our experience these disturbances are compensated by the benefit of non-flickering sequences.

Acknowledgements

The authors would like to express their appreciation to Prof. Heinrich Müller for reviewing this manuscript, and to Achim Stößer, who has designed the windmill-sequence.

Figure 6: Picture out of the sequence, which is now dithered with all refinements.

The threshold manipulation is dependent on the dilated graytone-change-picture. Old structures of previous pictures are no longer recognizable. By a closer view some minor irregularities in the pixel-patterns along the way the wing moved may be realized.

References

[ES86] J. Encarnacao and W. Straßer. *Computer Graphics.* Oldenbourg-Verlag, Mïchen, 1986.

[FS75] R.W. Floyd and L. Steinberg. An adaptive algorithm for spatial gray scale. *Int. Symp. Dig. Tech. Papers*, 36, 1975.

[Knu87] D.E. Knuth. Digital halftones by dot diffusion. *ACM Transactions on Graphics*, 6(4):245–273, Oktober 1987.

[Stu82] P. Stucki. Image processing for documentation. *Fachberichte und Referate*, Textverarbeitung und Bürosysteme(13):245–282, 1982.