Noname manuscript No.
(will be inserted by the editor)

Edge-aware inpainting with oriented (bi)harmonic interpolation

J.P. Lewis and Sebastian Sylwan

Abstract This short paper describes a simple approach to inpainting of small regions that may contain edges or other simple structure. Inpainting is commonly required in visual effect, both in order to remove fiducial markers that are commonly added for camera tracking purposes, and to paint out other small undesired objects. A body of sophisticated research on this subject has appeared over the last decade. However, few existing methods are suitable for motion-coherent inpainting of film or video. Our approach can propagate edges on the boundary into the interior of a spot in a temporally coherent way. The approach is also simple, requiring only the solution of sparse linear systems.

1 Introduction

In visual effects production it is often necessary to "erase" objects from a sequence of frames [6]. A particularly common case is the need to remove fiducial markers that are intentionally inserted to support camera tracking from the video. In other cases, it is necessary to remove wires, various rigs, blemishes, and so on.

A large body of research on this *inpainting* problem has developed over the last decade. Sophisticated methods can inpaint regions that have both structure and texture [2,5], although subtle artifacts are still visible in some cases. Inpainting algorithms have not been adopted in commercial VFX tools, and the industry relies on alternate (and more labour intensive) approaches such as spatial and temporal cloning.

For visual effects purposes, artifacts may be acceptable if they can only be identified by careful study of an image. A more serious problem is that of *temporal coherence*. As noted in [1] many existing methods result in obvious flickering or popping when applied to a video sequence, even if the individual frames are nearly seamless. This is particularly true of texture synthesis methods that involve selection (even random selection) of material from surrounding regions. Methods based on temporal texture synthesis have successfully addressed this problem, but these approaches can be brittle in their requirement of very similar image content elsewhere in the video and thus are most suited for textural regions. Other methods [4,7] assume that the region to be inpainted is isolated to a single or a few frames, as is the case with various types of noise and degradation.

This paper describes early work and initial results on a new approach this problem. We target the difficult case in which the occultation is present in all frames, as required in the visual effects tasks mentioned above. Our approach is an oriented extension of harmonic or biharmonic interpolation that can regenerate some edge structure in a temporally coherent way. Whereas many inpainting algorithms involve expensive iterative solution of a nonlinear PDE, ours involves only a few sparse linear system solves. The algorithm does not deal with textured regions, and has additional considerations that are indicated at the end of the paper.

2 Method

Our method is best introduced with a physical analogy. We imagine that image luminance is proportional to height, and a sheet of somewhat stiff material is pinned to the boundary of the missing region. The sheet will propagate edges on the boundary into the interior of the region.

J.P. Lewis
Weta Digital and Victoria University

Sebastian Sylwan
Weta Digital

In a digital simulation the sheet might be implemented by spline interpolation. However spline interpolation is defined as minimizing an approximate curvature (subject to constraints). A spline will thus minimize both the curvature across the hole (desired) and the curvature across the interpolated edge (undesired). Therefore, we seek a form of interpolation that is stiff only in the direction along the edge, as well as some way of identifying the edge direction.

The solution involves several simple steps. First, the hole is filled by spline interpolation. An edge on the boundary will extend inward, though it will begin to decay in the center of the region due to the curvature minimizing property just mentioned. However, this "edge-discovery" interpolation step is enough to estimate local gradient at each point in the interior of the region, which then gives a local direction for the edge, as well as a (inaccurate) measure of edge strength.

Using this, the hole is filled again, with an anisotropic form of harmonic or biharmonic interpolation. This second interpolation bends more easily across the edge, thus propagating the edge into the interior. If needed, the process can be iterated once or twice more.

These steps will now be described in more detail.

The initial edge-discovery interpolation problem is

$$\min_u \iint \left(u_{xx}^2 + 2u_{xy}^2 + u_{yy}^2\right) dxdy$$

subject to interpolating the derivative at the boundary.

In concept there are several approaches to solving this biharmonic scattered interpolation problem. Interpolation by radial basis functions is not preferred because of the very irregular data spacing, with dense points on the boundary and none in the interior of the hole. This intuition is made explicit with the notions of fill distance and separation distance [10]. The separation distance is related to the minimum eigenvalue of the RBF matrix and hence its condition number.

Instead, the interpolation is solved as a sparse linear system $\mathbf{Ax} = 0$ with $\mathbf{A}$ containing the stencil [8]

$$\begin{array}{ccccc} & & 1 & & \\ & 2 & -8 & 2 & \\ 1 & -8 & 20 & -8 & 1 \\ & 2 & -8 & 2 & \\ & & 1 & & \end{array}$$

This can be solved subject to hard constraints by moving the known values to the right hand side of the equation. Rather than implementing hard constraints directly, however, an equation

$$\alpha x = \alpha v_k$$

is added to the kth row for each constraint element v_k, resulting in the system

$$(\mathbf{A} + \alpha\mathbf{J})\mathbf{x} = \mathbf{b} + \alpha\mathbf{w}$$

where $\mathbf{w}$ is a vector containing the v_k (zero elsewhere) and $\mathbf{J}$ is diagonal matrix containing 1's in rows corresponding to constrained pixels. Taking α to infinity reproduces the constraint. We use a large, user-chosen number such as $\alpha = 10000$. This approach is chosen for two reasons: first, it simplifies the bookkeeping by preserving the original matrix dimensions. More importantly, it can accommodate boundaries larger than two pixels. This is helpful if we believe that the pixels are corrupted by noise, since it results in a least squares fit to the noisy, somewhat conflicting constraints.

The second step is estimation of the edge orientation at each point in the interpolated region. In concept this can be approached by estimating the principle curvature. In our experiments this performed unreliably at the scales of interest. Noting that first derivatives can be estimated more reliably than second derivatives in the presence of noise, we use the perpendicular to the gradient as an indicator of the edge direction.

The gradient direction is estimated using a standard computer vision approach, using the eigenvectors of the locally averaged "structure tensor"

$$\sum_{j \in N} w_j \sum_k^3 \nabla c_{j,k} \nabla c_{j,k}^T$$

where N is a neighborhood of the current pixel j, w_j are weights of a local mask, and $\nabla c_{j,k}$ is the gradient of the kth color channel at this pixel. The expression $\nabla c_k \nabla c_k^T$ for a single channel of a single pixel is a 2×2 symmetric matrix with a single non-zero eigenvalue reflecting the square of the gradient magnitude. The corresponding eigenvector is the gradient direction, and is perpendicular to the desired local edge direction. When this expression is summed over color channels and averaged over a neighborhood, generally both eigenvalues will be non-zero. The ratio of the eigenvalues reflects how "oriented" or edge-like the region is, and the leading eigenvector is the dominant gradient direction over the region. With λ_0, λ_1 being the smaller and larger eigenvalue respectively, the measure $e = 1 - (\lambda_0 + \epsilon)/(\lambda_1 + \epsilon)$ describes orientedness as a number between 0 and 1. ϵ is a small regularizing constant. The neighborhood size is a parameter that should be set by the operator. It should be as small as possible, but large enough to get a reliable edge estimate.

Given the edge direction and orientedness e, the last step is to perform an edge-aware interpolation using a

locally-varying oriented version of the Laplacian or iterated Laplacian. The formulation of this step borrows from anisotropic diffusion.

Since the Laplacian is the trace of the Hessian

$$\mathbf{H} = \begin{bmatrix} \partial_{xx} & \partial_{xy} \\ \partial_{xy} & \partial_{yy} \end{bmatrix}$$

an axis-aligned oriented Laplacian can be produced by $\operatorname{tr} \mathbf{S}^T \mathbf{H} \mathbf{S}$ where $\mathbf{S}$ is a diagonal scaling matrix. The entries in $\mathbf{S}$ are $1, e$. An arbitrarily oriented and elongated Laplacian ∇^2_O results from applying a rotation $\mathbf{Q}$:

$$\nabla^2_O = \operatorname{tr} \mathbf{S}^T \mathbf{Q}^T \mathbf{H} \mathbf{Q} \mathbf{S}$$

An alternate expression used in the anisotropic literature [9] generalizes the equality $\nabla^2 u = \nabla \cdot \nabla u$ to

$$\nabla^2_O u = \nabla \cdot \mathbf{T} \nabla u$$

where $\mathbf{T} = \mathbf{Q} \mathbf{S} \mathbf{S}^T \mathbf{Q}^T$. The equivalence of the two forms follows from the rule $\operatorname{tr} ab^T = a^T b$.

The resulting oriented Laplacian is

$$\nabla^2_O = a\partial_{xx} + 2b\partial_{xy} + c\partial_{yy}$$

where a, b, c are the unique elements of $\mathbf{T}$. Discretizing this with finite differences results in a per-pixel stencil that can be plugged into a sparse linear system solve.

The resulting interpolation is often suitable in our (relatively limited) experience. However, it can be improved in some cases by re-estimating the edge direction using the result of the oriented interpolation. We have seen slight improvement with up to three iterations.

Laplacian interpolation can be interpreted as minimizing stretching, whereas biharmonic interpolation can be interpreted as minimizing an approximate curvature. Denoting the oriented Laplacian system as

$$(\mathbf{L}_O + \alpha \mathbf{J})\mathbf{x} = \mathbf{b} + \alpha \mathbf{w},$$

oriented biharmonic interpolation can be obtained by replacing the oriented Laplacian matrix $\mathbf{L}_O$ with $\mathbf{L}_O^T \mathbf{L}_O$. We believe that the oriented biharmonic solution is more suitable for our goals, however, it can also amplify noise in the boundary conditions. In other contexts it has proven useful to blend these two energies (e.g. [3]), so this choice is provided to the artist.

As described the algorithm has several operator-adjusted parameters:

- the data fidelity α,
- the orientation regularizing constant ϵ. This is currently set at 1; it may be necessary to increase it if the edge estimation is unreliable. A situation that might require this is an image area that is essentially flat in luminance but corrupted by strong noise, such as the blue channel of film in a dark region,
- the choice of harmonic or biharmonic interpolation,
- the option of iterating the solution using edges estimated from a previous oriented interpolation step,
- the size of the neighborhood used in estimating the edge direction and strength. In fact we allow the operator to select a smaller size for the second iteration. The images in this paper were made with widths between 5 and 30.

3 Results

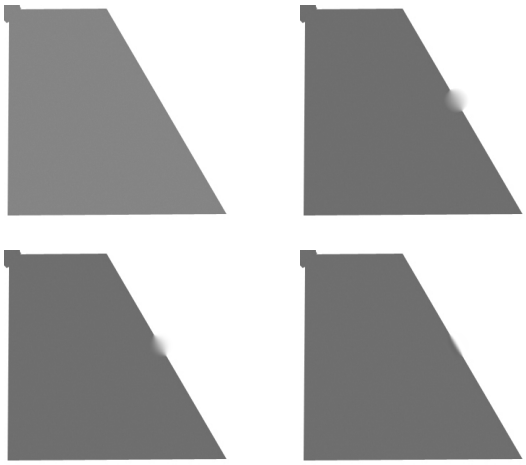

Fig. 1 Top left: original image. Top right: Laplace interpolation. Bottom left: biharmonic interpolation. Bottom right: oriented harmonic interpolation.

Fig. 1 shows the result of interpolating an edge using the oriented Laplacian solve. A slight blurring of the edge is visible on enlargement, however the result is significantly better than standard Laplace interpolation.

Fig. 2 shows the result of interpolating part of a zebra. Here as well, some blurring is visible on enlargement. Note that the inpainted region here is larger than that shown in many competing methods.

Fig. 3 shows a frame from a video example in which a tracking marker has been inpainted. In this section of the video sequence the marker aligns with a highlight, requiring the use of oriented inpainting. Although the algorithm is applied independently at each frame, the result is temporally coherent. This result is somewhat intuitive considering the physical analogies for this type of interpolation (stretched and bent sheets): as long as the boundary is itself temporally coherent, the interpolation should be as well.

Fig. 4 shows the a more challenging case in which a portion of an airplane is removed in a video shot.

Several observers were not able to guess the location of the inpainted region, although it can be seen under enlargement once pointed out. The inpainted result is temporally coherent.

4 Discussion

The method described here requires only linear algebra and a sparse linear system solver. It is fast, requiring roughly 0.03 seconds for the zebra example (a video-resolution image) when run a single core of a current high-end PC. We use the SuperLU sparse direct solver, so expect $O(n^3)$ performance in the number of filled pixels n.

The algorithm is limited in that it cannot handle textured regions, and temporal coherence can be achieved only if the boundary of the inpainted region is temporally smooth. It is thus more suited for operation on man-made objects than on natural scenes. On the other hand, "texture" involves a concept of scale. Our algorithm would not succeed in inpainting a region of tree bark, yet it might succeed in inpainting a small marker that is fixed to the same tree.

A final consideration is that it exposes several parameters that should be tuned to the particular imagery. The algorithm is thus suitable for visual effects work, where the operators are skilled artists, but less suitable for casual or automatic applications.

5 Acknowledgements

We thank Eitan Grinspun and Pravin Bhat for discussion of several points.

References

1. Aubert, G., Kornprobst, P.: Mathematical problems in image processing: partial differential equations and the calculus of variations. Springer (2006)
2. Bertalmio, M., Vese, L., Sapiro, G., Osher, S.: Simultaneous structure and texture image inpainting. Image Processing, IEEE Transactions on **12**(8), 882–889 (2003)
3. Bickel, B., Botsch, M., Angst, R., Matusik, W., Otaduy, M., Pfister, H., Gross, M.: Multi-scale capture of facial geometry and motion. ACM Trans. Graph. **26**(3), 33 (2007). DOI http://doi.acm.org/10.1145/1276377.1276419
4. D'Amore, L., Marcellino, L., Murli, A.: Image sequence inpainting: towards numerical software for detection and removal of local missing data via motion estimation. J. Comput. Appl. Math. **198**, 396–413 (2007)
5. Fadili, M., Starck, J., Murtagh, F.: Inpainting and zooming using sparse representations. The Computer Journal **52**(1), 64 (2009)
6. Kokaram, A.: Motion picture restoration: digital algorithms for artefact suppression in degraded motion picture film and video. Springer-Verlag (1998)
7. Lauze, F., Nielsen, M.: A variational algorithm for motion compensated inpainting. In: British Machine Vision Conference, vol. 2, pp. 777–787 (2004)
8. Szeliski, R.: Bayesian modeling of uncertainty in low-level vision. Kluwer Academic Publishers (1989)
9. Weickert, J.: Anisotropic diffusion in image processing, vol. 256. Teubner-Verlag (1998)
10. Wendland, H.: Scattered Data Approximation. Cambridge (2005)

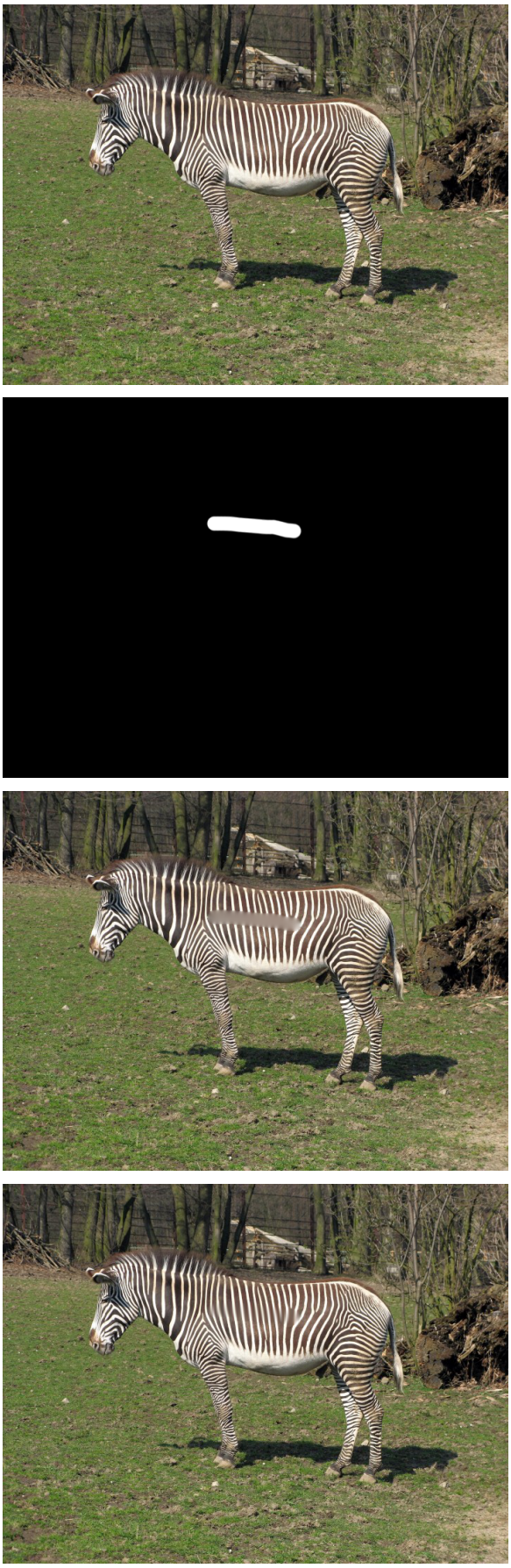

Fig. 2 Top: original image and mask. Bottom left: Laplace interpolation. Bottom right: oriented biharmonic interpolation. Please enlarge to see details.

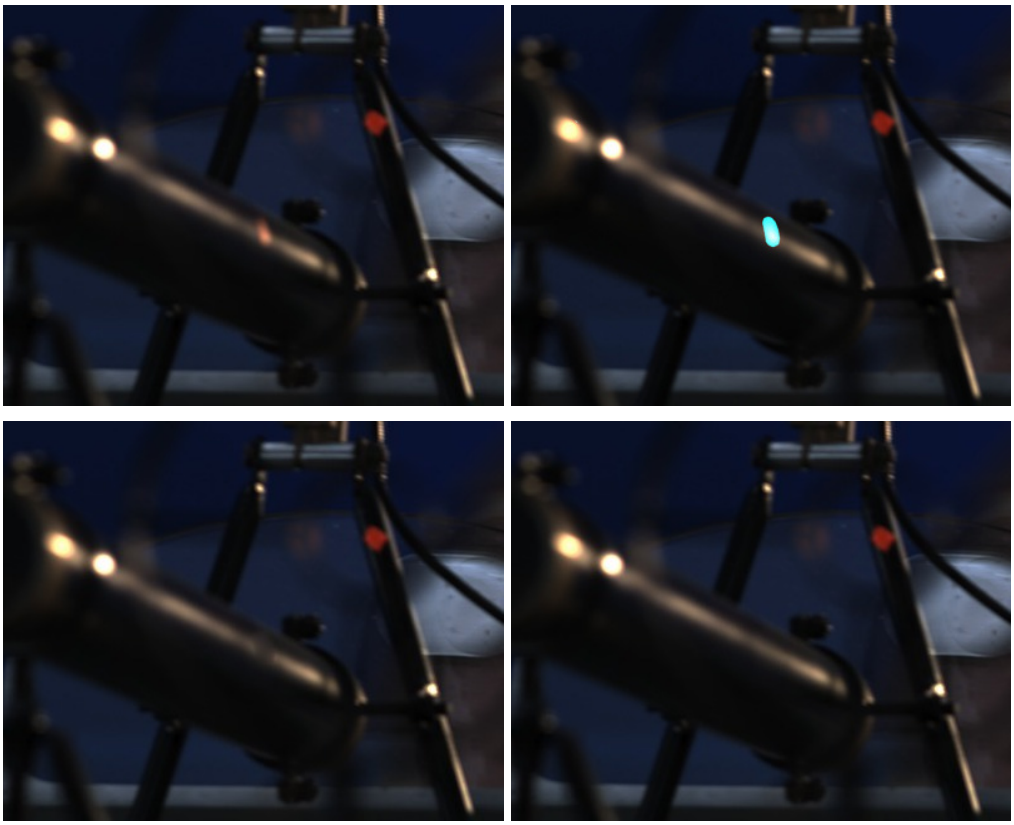

Fig. 3 Oriented inpainting on a video example. Top left: original image. Top right: marker detected by a semi-automatic classification process. Bottom left: harmonic interpolation. Bottom right: oriented biharmonic interpolation. Please enlarge to see details (also please refer to the video accompanying the paper).

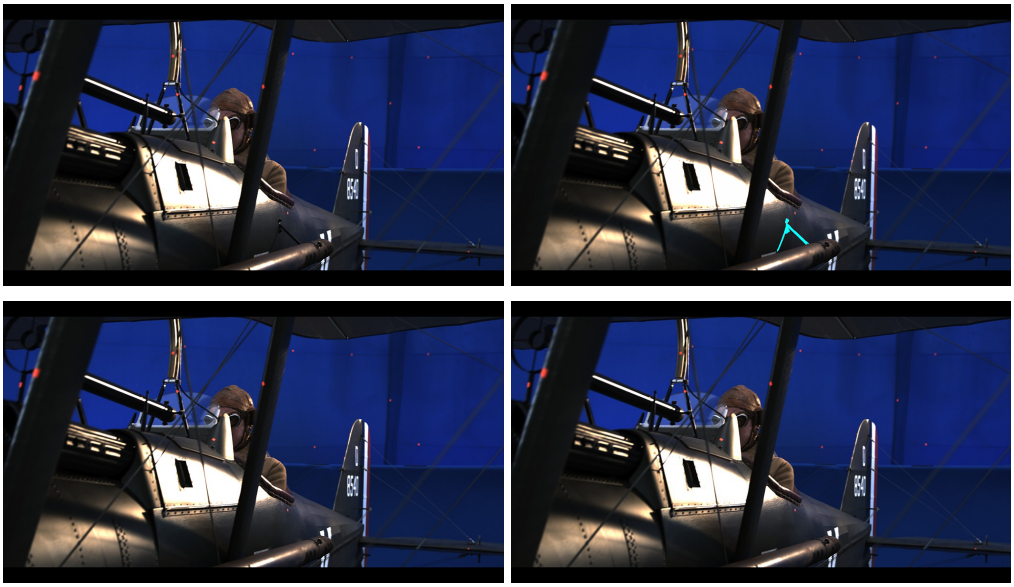

Fig. 4 A second video example in which oriented inpainting is required. Top left: original image. Top right: structure detected by a semi-automatic classification process. Bottom left: biharmonic interpolation. Bottom right: oriented biharmonic interpolation. The oriented inpainting produces a better result in the region at the right bottom, where the inpainting crosses an occluding edge. Please enlarge to see details (also please refer to the video accompanying the paper).