Description

Image inpainting is a form of **image conservation** and **image restoration,**

* Restore old, degraded photos
* Repair photos with missing areas due to damage and aging
* Mask out and remove particular objects from an image (and do so in an aesthetically pleasing way)

Based on FMM

Most of you will have some old degraded photos at your home with some black spots, some strokes etc on it. We can’t simply erase them in a paint tool because it is will simply replace black structures with white structures which is of no use. In these cases, a technique called image inpainting is used. The basic idea is simple: Replace those bad marks with its neighbouring pixels so that it looks like the neigbourhood.

Digital inpainting

serves a wide range of applications, such as removing text and logos from

still images or videos, reconstructing scans of deteriorated images by removing

scratches or stains, or creating artistic effects.

What is FMM

‘**An Image Inpainting Technique Based on the Fast Marching Method”** by Alexandru Telea in 2004.

 It is based on Fast Marching Method. Consider a region in the image to be inpainted. Algorithm starts from the boundary of this region and goes inside the region gradually filling everything in the boundary first. It takes a small neighbourhood around the pixel on the neigbourhood to be inpainted. This pixel is replaced by normalized weighted sum of all the known pixels in the neigbourhood. Selection of the weights is an important matter. More weightage is given to those pixels lying near to the point, near to the normal of the boundary and those lying on the boundary contours. Once a pixel is inpainted, it moves to next nearest pixel using Fast Marching Method. FMM ensures those pixels near the known pixels are inpainted first, so that it just works like a manual heuristic operation. This algorithm is enabled by using the flag,

To explain our use of the FMM in detail–and since the FMM is not

straightforward to implement from the reference literature [Sethian 96, Sethian

99]–we provide next its complete pseudocode. The FMM maintains a

so-called narrow band of pixels, which is exactly our inpainting boundary ∂Ω.

For every image pixel, we store its value T, its image gray value I (both

represented as floating-point values), and a flag f that may have three values:

• BAND: the pixel belongs to the narrow band. Its T value undergoes

update.

• KNOWN: the pixel is outside ∂Ω, in the known image area. Its T and

I values are known.

• INSIDE: the pixel is inside ∂Ω, in the region to inpaint. Its T and I

values are not yet known.

The FMM has an initialization and propagation phase as follows. First,

we set T to zero on and outside the boundary ∂Ω of the region to inpaint

and to some large value (in practice 106) inside, and initialize f over the

whole image as explained above. All BAND points are inserted in a heap

30 journal of graphics tools

while (NarrowBand not empty)

{

extract P(i,j) = head(NarrowBand); /\* STEP 1 \*/

f(i,j) = KNOWN;

for (k,l) in (i1,j),(i,j1),(i+1,j),(i,j+1)

if (f(k,l)!=KNOWN)

{

if (f(k,l)==INSIDE)

{

f(k,l)=BAND; /\* STEP 2 \*/

inpaint(k,l); /\* STEP 3 \*/

}

T (k,l) = min(solve(k1,l,k,l1), /\* STEP 4 \*/

solve(k+1,l,k,l1),

solve(k1,l,k,l+1),

solve(k+1,l,k,l+1));

insert(k,l) in NarrowBand; /\* STEP 5 \*/

}

}

float solve(int i1,int j1,int i2,int j2)

{

float sol = 1.0e6;

if (f(i1,j1)==KNOWN)

if (f(i2,j2)==KNOWN)

{

float r = sqrt(2(T(i1,j1)T(i2,j2))\*(T(i1,j1)T(i2,j2)));

float s = (T(i1,j1)+T(i2,j2)r)/2;

if (s>=T(i1,j1) && s>=T(i2,j2)) sol = s;

else

{ s += r; if (s>=T(i1,j1) && s>=T(i2,j2)) sol = s; }

}

else sol = 1+T(i1,j1));

else if (f(i2,j2)==KNOWN) sol = 1+T(i1,j2));

return sol;

}

Figure 4. Fast marching method used for inpainting.

NarrowBand sorted in ascending order of their T values. Next, we propagate

the T, f, and I values using the code shown in Figure 4. Step 1 extracts

the BAND point with the smallest T. Step 2 marches the boundary inward

by adding new points to it. Step 3 performs the inpainting (see Section 2.3).

Step 4 propagates the value T of point (i, j) to its neighbors (k, l) by solving

Telea: An Image Inpainting Technique 31

the finite difference discretization of Equation 3 given by

max(D−xT, −D+xT, 0)2 +max(D−yT, −D+yT, 0)2 = 1, (4)

where D−xT(i, j) = T(i, j) − T(i − 1, j) and D+xT (i, j) = T(i+1, j) − T (i, j)

and similarly for y. Following the upwind idea of Sethian [Sethian 96], we

solve Equation 4 for (k, l)’s four quadrants and retain the smallest solution.

Finally, Step 5 (re)inserts (k, l) with its new T in the heap.

Comparision with other

We prefer the FMM over other Distance Transform (DT) methods that

compute the distance map T to a boundary ∂Ω (e.g., [Borgefors 84, Borgefors

86, Meijster et al. 00]). The FMM’s main advantage is that it explicitly

maintains the narrow band that separates the known from the unknown image

area and specifies which is the next pixel to inpaint. Other DT methods

compute the distance map T but do not maintain an explicit narrow band.

Adding a narrow band structure to these methods would complicate their

implementation, whereas the FMM provides this structure by default.