Fast Marching farthest point sampling

Carsten Moenning and Neil A. Dodgson

University of Cambridge, Computer Laboratory, 15 JJ Thomson Avenue, Cambridge CB3 0FD - UK, {cm230,nad}@cl.cam.ac.uk

Abstract

We introduce the Fast Marching farthest point sampling (FastFPS) approach for the progressive sampling of planar domains and curved manifolds in triangulated, point cloud or implicit form. By using Fast Marching methods[2, 3, 6] for the incremental computation of distance maps across the sampling domain, we obtain a farthest point sampling technique superior to earlier point sampling principles in two important respects. Firstly, our method performs equally well in both the uniform and the adaptive case. Secondly, the algorithm is applicable to both images and higher dimensional surfaces in triangulated, point cloud or implicit form. This paper presents the methods underlying the algorithm and gives examples for the processing of images and triangulated surfaces. A companion report[4] provides details regarding the application of the FastFPS algorithm to point clouds and implicit surfaces.

Categories and Subject Descriptors (according to ACM CCS): I.3.5 [Computer Graphics]: Computational Geometry and Object Modelling; I.4.5 [Computer Graphics]: Image Processing and Computer Vision

1. Introduction

We consider the problem of sampling progressively from planar domains or curved manifolds in triangulated, point cloud or implicit form. An efficient solution to this problem is of interest for a large number of applications including progressive transmission, point-based multiresolution representation and implicit surface rendering, etc. The method may further be used for the efficient uniform or feature-sensitive simplification of both images and 3D surfaces in triangulated, implicit or point cloud form. In the case of surfaces in point cloud or implicit form, this is achieved without the need for any prior surface reconstruction.

Eldar et al.[1] introduce an efficient uniform irregular "farthest point" (image) sampling strategy featuring a high data acquisition rate, excellent anti-aliasing properties and an elegant relationship to the Voronoi diagram[5] concept. Similar to other point sampling techniques, however, Eldar et al.[1] farthest point approach is restricted to planar domains and does not extend to the case of non-uniform sampling. We propose an alternative farthest point technique which incrementally constructs discrete Voronoi diagrams across planar domains or higher dimensional surfaces with the help of Fast Marching methods[2, 3, 6]. This novel approach yields a very efficient implementation with the resulting Voronoi diagrams remaining tractable even when modelling a non-uniform metric on surfaces in point cloud or implicit form and without the need for any prior surface reconstruction.

We briefly discuss the relevant farthest point sampling and Fast Marching concepts, followed by our farthest point sampling algorithm and brief examples for the processing of images and triangulated surfaces. A companion technical report[4] provides details regarding the FastFPS algorithm for point clouds and implicit surfaces.

2. Previous Work

2.1. Farthest point sampling

Farthest point sampling is based on the idea of repeatedly placing the next sample point in the middle of the least-known area of the sampling domain. In the following, we summarise the reasoning underlying this approach presented in Eldar et al.[1]

Starting with the uniform case, the authors consider the case of an image representing a continuous stochastic process featuring constant first and second order central moments with the third central moment, i.e., the covariance, decreasing exponentially with spatial distance. This assumption of stationary first and second order central moments

leads to the result that the expected mean square (reconstruction) error, ε^2, depends on the location of the $N+1$th sample only

$$\varepsilon^2(p_0,\ldots,p_{N-1}) = \int\int \sigma^2 - U^T R^{-1} U \, dx\, dy \qquad (1)$$

where (covariance matrix)

$$R_{ij} = \sigma^2 e^{-\lambda\sqrt{(x_i-x_j)^2+(y_i-y_j)^2}}$$

and (variance matrix)

$$U_i = \sigma^2 e^{-\lambda\sqrt{(x_i-x)^2+(y_i-y)^2}}$$

for all $0 \leq i, j \leq N-1$. Since stationarity implies that the image's statistical properties are spatially invariant and given that point correlations decrease with distance, uniformly choosing the $N+1$th sample point to be that point which is farthest away from the current set of sample points therefore represents the optimal sampling approach within this framework.

This sampling strategy is intimately linked with the incremental construction of a Voronoi diagram over the image domain. To see this, note that the point farthest away from the current set of sample sites, S, is represented by the centre of the largest circle empty of any site $s_i \in S$. Eldar et al.[1] show that in the case of farthest point sample sets, the centre of such a circle is given by a vertex of the bounded Voronoi diagram of S, $BVD(S)$. Thus, incremental (bounded) Voronoi diagram construction provides farthest point sample points progressively.

From visual inspection of images it is clear that usually not only the sample covariances but also the sample means and variances vary spatially across an image. When allowing for this more general variability, the design of a non-uniform, adaptive sampling strategy is required. In this context, the assumption of sample point covariances decreasing, exponentially or otherwise, with point distance remains valid. However, since continuous Voronoi diagrams in non-uniform metrics may lose favourable properties such as connected and convex regions[5], Eldar et al.[1] are led to conclude that finding the farthest point in such a diagram is impractical. They opt for a work-around involving the application-dependent weighting of the vertices in the uniform Euclidean Voronoi diagram instead. In this paper, we put forward a Fast Marching-based farthest point sampling approach which works equally well in both the uniform and adaptive case. Furthermore, unlike previous point sampling techniques, it is not limited to 2D domains but extends to higher dimensional surfaces in triangulated, point cloud or implicit form without any loss in efficiency or the need for prior surface reconstruction.

2.2. Fast Marching

We pose the problem of computing the distance map across a sampling domain in the form of a specific boundary value partial differential equation and briefly outline the Fast Marching approach towards the very efficient approximation of its solution.

For simplicity, take the case of an interface propagating with speed function $F(x,y)$ away from a source (boundary) point (u,v) across a planar Euclidean domain. When interested in the time of arrival, $T(x,y)$, of the interface at grid point (x,y), i.e., the distance map T given source point (u,v), the relationship between the magnitude of the distance map's gradient and the given weight $F(x,y)$ at each point can be expressed as the following boundary value formulation

$$|\nabla T(x,y)| = F(x,y) \qquad (2)$$

with boundary condition $T(u,v) = 0$. That is, the distance map gradient is proportional to the weight function. The problem of determining a weighted distance map has therefore been transformed into the problem of solving a particular type of Hamilton-Jacobi partial differential equation, the Eikonal equation. For $F(x,y) > 0$, this type of equation can be solved for $T(x,y)$ using Fast Marching.

Since the Eikonal equation is well-known to become non-differentiable through the development of corners and cusps during propagation[6], the Fast Marching method considers only upwind, entropy-satisfying finite difference approximations to the equation. For such first and second order approximations, see Sethian[6].
The use of an upwind difference approximation implies that information propagates from smaller to larger values of T only, i.e., a grid point's arrival time gets updated by neighbouring points with smaller T values only. This monotonicity property allows for the maintenance of a narrow band of candidate points around the front representing its outward motion. The property can further be exploited for the design of a simple and efficient algorithm by freezing the T values of existing points and subsequently inserting neighbouring ones into the narrow band thereby marching the band forward. Arrangement of the band elements in a min-heap leads to an $O(N \log N)$ implementation, with N representing the number of grid points.

The algorithm can relatively easily be extended to the case of surfaces in triangulated, point cloud or implicit form. For the case of triangulated domains, suitable upwind approximations are presented in Sethian[6]. As regards surfaces in point cloud or implicit surface form, Mémoli and Sapiro[2, 3] present an augmentation of the Fast Marching method for the computation of distance functions across point clouds or implicit surfaces without the need for any prior surface reconstruction. This technique is exploited in Moenning and Dodgson[4] for the extension of the FastFPS principle to surfaces in point cloud or implicit form.

3. Fast Marching farthest point sampling

For simplicity, we first consider our FastFPS algorithm for a uniform metric and a planar domain.

FastFPS for planar domains Starting with an initial sample point set S, we compute $BVD(S)$ by simultaneously propagating fronts from each of the initial sample points outwards. This process is equivalent to the computation of the Euclidean distance map across the domain given S and is achieved by solving the Eikonal equation with $F(x,y) = 1$, for all x, y, and using a single min-heap. The vertices of $BVD(S)$ are given by those grid points entered by three or more propagation waves (or two for points on the domain boundary) and are therefore obtained as a by-product of the propagation process. The Voronoi vertices' arrival times are inserted into a max-heap data structure. The algorithm then proceeds by extracting the root from the max-heap, the grid location of which represents the location of the next farthest point sample. The sample is inserted into $BVD(S)$ by resetting its arrival time to zero and propagating a front away from it. The front will continue propagating until it hits grid points featuring lower arrival times and thus belonging to a neighbouring Voronoi cell. The T values of updated grid points are updated correspondingly in the min-heap using back pointers. New and obsolete Voronoi vertices are inserted or removed from the max-heap respectively. The algorithm continues extracting the root from the max-heap until it is empty or the sample point budget has been exhausted. By allowing $F(x,y)$ to vary with any weights associated with points in the domain, this algorithm is easily extended to the case of adaptive sampling. The algorithm can thus be summarised as follows

0) Given an initial sample set S, $n = |S| \geq 2$, compute $BVD(S)$ by propagating fronts with speed $F(x,y)$ from the sample points outwards using a min-heap and Fast Marching with a finite difference approximation for planar domains[6]. Store the Voronoi vertices' arrival times in a max-heap.
1) Extract the root from the max-heap to obtain s_{n+1}. $S' = S \cup \{s_{n+1}\}$. Compute $BVD(S')$ by propagating a front locally from s_{n+1} outwards using the min-heap and Fast Marching as in 0).
2) Correct the arrival times of updated grid points in the min-heap. Insert the vertices of the new Voronoi polygon, $V(s_{n+1}, S')$, in the max-heap. Remove obsolete Voronoi vertices of the neighbours of $V(s_{n+1}, S')$ from the max-heap.
3) If neither the max-heap is empty nor the point budget has been exhausted, loop from 1).

Extracting the root from, inserting into and removing from the max-heap with subsequent re-heapifying are $O(\log M)$ operations, where M represents the number of elements in the heap. M is $O(N)$, N representing the number of grid points. The updating of existing min-heap entries is $O(1)$ due to the use of back pointers from the grid to the heap. The detection of a (bounded) Voronoi cell's vertices and boundary is a by-product of the $O(N \log N)$ front propagation. Thus, the algorithm's worst case running time is $O(N \log N)$.

FastFPS for triangulated domains The algorithm necessarily no longer considers points in an orthogonal grid but vertices, v_i, $i = 1,2,\ldots,N$, in a triangulated domain. Front propagation occurs directly on the surface with F being a positive constant (uniform) or varying with any cost associated with the vertices (non-uniform). A suitable monotone and consistent finite difference approximation for triangulated domains can be found in Sethian[6]. To allow for triangulated domains, only steps 0) and 1) of the FastFPS algorithm need to be modified as follows

0) Given an initial sample set S, $n = |S| \geq 2$, compute $BVD(S)$ by propagating fronts with speed $F(v_i)$ from the sample points outwards using a min-heap and Fast Marching with a finite difference approximation for triangulated domains. March along the triangles and linearly interpolate the intersection curve between pairs of distance maps of different origin across each triangle[6]. Store the Voronoi vertices' arrival times in a max-heap.
1) Extract the root from the max-heap to obtain s_{n+1}. $S' = S \cup \{s_{n+1}\}$. Compute $BVD(S')$ by propagating a front locally from s_{n+1} outwards as in 0). March the triangles touched by this local update procedure and interpolate the intersection curves.

Triangle marching and the linear interpolation of the intersection curves are $O(N)$ processes, N representing the number of vertices. The remaining operations are as in FastFPS for planar domains so the overall complexity of FastFPS for triangulated domains is $O(N \log N)$.

Model	*Model size*	*Sample size*	*Secs.*
LENA image	(256x256)	30k	1.21
MANDRILL image	(348x348)	40k	2.45
PEPPERS image	(512x512)	50k	4.24
BUNNY mesh	35947	30k	0.72
DAVID mesh	99455	40k	1.43
DRAGON mesh	184018	50k	3.89

Table 1: Uniform FastFPS sampling times.

4. Application examples

Starting with the application of FastFPS for planar domains to progressive image sampling, we adaptively sample the (512x512) Lena image by making $F(x,y)$ vary with a similarity measure. More specifically, we transform RGB values into CIELAB colour space coordinates and impose a non-uniform speed function in the form of inter-point colour space distances. Like any other weight function, this similarity measure can be easily incorporated into a FastFPS implementation provided $F(x,y) > 0$, for all x,y. See figure 1 for

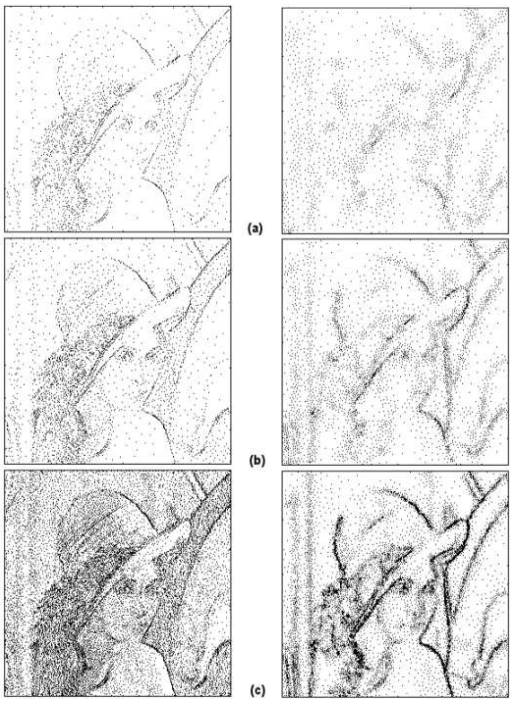

Figure 1: *The adaptive FastFPS point set (left) captures vital features very well early on into the sequence and without the excessive concentration of points near sharp edges produced by Eldar et al.*[1] *algorithm (right).*
Sample size: (a) 0.8% (2k), (b) 1.6% (4k), (c) 6.1% (16k).

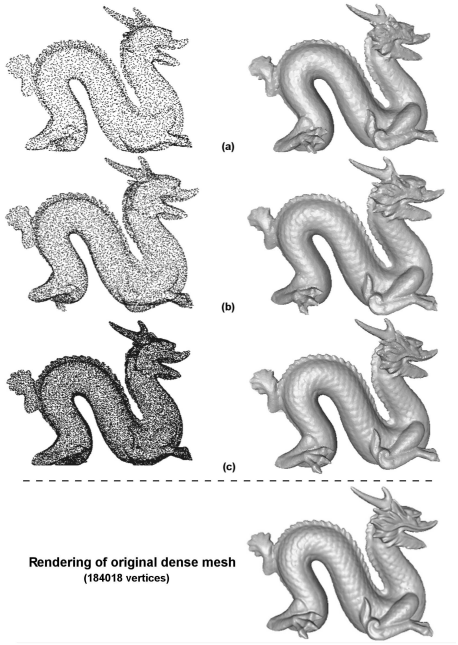

Figure 2: *Points sets (left) produced by uniform FastFPS for triangulated domains and the corresponding mesh renderings (right).*
Sample size: (a) 4.3% (8k), (b) 8.6% (16k), (c) 21.7% (40k)

the high-quality point sets produced by FastFPS for planar domains for relatively small sample point budgets.

We apply uniform FastFPS for triangulated domains to the problem of sampling the "Dragon" object surface for mesh decimation and/or progressive transmission. The FastFPS point sets for different budgets are presented in figure 2. The cluster-free point sets fill the space uniformly and irregularly thereby suppressing any noticeable aliasing effects and allowing for both high-quality renderings early on into the sequence and significantly decimated model sizes. Although a thorough experimental analysis of execution (and memory) efficiency will have to be provided elsewhere, table 1 gives an indication of the algorithm's speed on a AMD 1.3 Ghz, 256MB, Windows 2000 machine.

5. Conclusions

We presented a new, efficient and easily implementable Fast Marching-based[2, 3, 6] farthest point sampling approach. Its main benefits are, firstly, that it works equally well in the uniform and adaptive case. Secondly, FastFPS is applicable to planar domains and surfaces in triangulated, point cloud or implicit form. We demonstrated the quality and speed of computation of FastFPS point sets for the generation of effective sparse image and 3D surface representations.

References

1. Y. Eldar, M. Lindenbaum, M. Porat and Y. Y. Zeevi. The Farthest Point Strategy for Progressive Image Sampling. *IEEE Transactions on Image Processing*, **6**(9):1305–1315, 1997.

2. F. Mémoli and G. Sapiro. Fast Computation of Weighted Distance Functions and Geodesics on Implicit Hyper-Surfaces. *Journal of Computational Physics*, **173**(1):764–795, 2001.

3. F. Mémoli and G. Sapiro. Distance Functions and Geodesics on Point Clouds. *Technical Report 1902, IMA, University of Minnesota*, USA, 2002.

4. C. Moenning and N. A. Dodgson. Fast Marching farthest point sampling for implicit surfaces and point clouds. *Technical Report 565, Computer Laboratory, University of Cambridge*, UK, 2003.

5. A. Okabe, B. Boots and K. Sugihara. *Spatial Tessellations*. 2nd ed. John Wiley, Chicester, UK, 2000.

6. J. A. Sethian. *Level Set Methods and Fast Marching Methods*. 2nd ed. Cambridge University Press, Cambridge, UK, 1999.