Efficient and Flexible Sampling with Blue Noise Properties of Triangular Meshes

Massimiliano Corsini, Paolo Cignoni, and Roberto Scopigno

Abstract—This paper deals with the problem of taking random samples over the surface of a 3D mesh describing and evaluating efficient algorithms for generating different distributions. We discuss first the problem of generating a Monte Carlo distribution in a efficient and practical way avoiding common pitfalls. Then, we propose *Constrained Poisson-disk* sampling, a new Poisson-disk sampling scheme for polygonal meshes which can be easily tweaked in order to generate customized set of points such as importance sampling or distributions with generic geometric constraints. In particular, two algorithms based on this approach are presented. An in-depth analysis of the frequency characterization and performance of the proposed algorithms are also presented and discussed.

Index Terms—Geometry Processing, Computational Geometry, Three-Dimensional Graphics and Realism, Sampling, Poisson-disk sampling, Monte Carlo methods

1 Introduction

Sampling is a fundamental operation in many areas, in particular in Computer Graphics, where several rendering and geometry processing algorithms use different sampling schemes depending on the specific purpose. In these last years a lot of research effort has been envisaged to develop new and efficient sampling algorithms, especially to produce sampling with blue noise characteristics, that best fit the needs of several geometric modeling and graphics applications [1], [2], [3]. Despite these efforts not so much attention has been paid to the development of algorithms that generate samples directly on a 3D surface, in particular on a generic triangular mesh. Research on remeshing implicitly account for surface mesh sampling, but most of the proposed techniques focus on the specific task of remeshing and are not general. Our aim is to investigate the issue of sampling meshes in an efficient and flexible way ranging from uniform sampling, to Poisson-disk sampling, passing to scheme with blue noise properties with geometric constraints. In this context, we discuss efficient uniform sampling of triangular meshes and propose two algorithms based on a novel method for the fast generation of Poisson-disk distributions on meshes.

The rationale behind discussing Monte Carlo sampling algorithms, that could be considered as a trivial problem, is the sake of highlighting common pitfalls, for example how to avoid biased distributions. For this purpose we describe the risk of a straightforward implementation and we illustrate two approaches for the two different needs of a triangle-by-triangle sampling or of a continuous stream of samples all over the mesh. Moreover, our Poisson-disk sampling algorithms are based on this generation.

Poisson-disk sampling is one of the most common sampling schemes in the context of Computer Graphics due to its blue noise properties. It consists of generating a uniformly random distribution where the minimum distance between each sample is $2r$; therefore a disk of radius r centered on each sample does not overlap any other disk. This guarantees that objects of a certain size can be distributed according to this sampling scheme without overlapping. Moreover, this type of sampling has been demonstrated to be particularly suitable in ray tracing and other rendering algorithms for its blue noise characteristics [4], [5]. For these reasons several algorithms for Poisson-disk sampling of the planar domain have been developed in the last years. Concerning the case of 3D surface represented as meshes, according to our knowledge the only works of this kind are the ones of Li et al. [1], Fu et al. [6] and Cline et al. [3] and the recent [7]. Here, we propose to sample the mesh surface by means of an extension of the hierarchical approach of White et al. [8]. Additionally, we will show that our Poisson-disk method is able to generate importance sampling with trivial modifications and, more important, it can be easily tweaked to generate *constrained Poisson-disk* distributions, i.e. Poisson-disk like distributions with geometric constraints on sample placement. This makes the proposed sampling scheme very flexible and effective in different application contexts.

The effectiveness of the proposed algorithms has been tested using the de-facto standard frequency analysis of sampling patterns, i.e. the *radial average power* and the *radial anisotropy*. The *relative radius coefficient* (ρ) has been also evaluated [2]. With these sets of experimental results we will show also that the proposed sampling algorithms are completely independent of the mesh connectivity like any good sampling algorithm for polygonal

• *Massimiliano Corsini, Paolo Cignoni and Roberto Scopigno are with the Visual Computing Lab, ISTI-CNR, Pisa, Italy.*
E-mail: firstname.lastname@isti.cnr.it

meshes should be.

The contribution of this paper can be summarized as follows:

- A discussion of some implementation issues related to Monte Carlo sampling of triangular meshes.
- A novel approach for the generation of what we call *constrained Poisson-disk* distribution, i.e. a Poisson-disk like distribution that could include also additional constraints.
- Two Poisson-disk sampling algorithms for 3D meshes, based on this novel approach, that are independent of the number of mesh elements, mesh connectivity and mesh triangles' shape.

Moreover, we already made publicly available a robust implementation of all the presented algorithms; we have integrated, since 2009, all the discussed techniques within the open source mesh processing system MeshLab [9].

2 Related Work

The literature on sampling is huge and extensive. Here, we discuss the works closer to our aims; in particular we focus on Poisson-disk sampling.

2.1 Poisson-Disk Sampling on Planar Domain

One of the first algorithms for the generation of Poisson-disk sampling is Dart-Throwing [4]. This very simple algorithm randomly generates a sample and discards it if the minimum distance constraint is not respected.

A class of algorithms for Poisson-disk distribution generation is based on tiling. The basic idea is to pregenerate samples inside tiles (e.g. using dart-throwing) and to fill the sampling domain with the pregenerated tiles in order to produce the desired distribution. Wang Tiles are often employed [10], [11] to obtain non-periodic tiling. Lagae and Dutrè [12] derived a variant of Wang Tiles more suitable for Poisson sampling called *dual tiling*. They extend this idea also to generate Poisson spheres distributions in the 3D space [13]. Even if this method works in 3D space it is not directly applicable to sample 3D meshes. In order to control the density of the sampling Ostromoukhov et al. [14] employed Penrose tiling of the plane; a subdivision rule for Penrose tiling is proposed to refine the sampling where high density is required.

Dunbar and Humphreys [15] proposed a very efficient algorithm with $O(N \log N)$ complexity where N is the number of samples to generate. The main idea is to compute and encode the remaining portion of the domain that still does not violate the distance constraint each time a sample is generated, and to use only this valid domain for the successive sample generation. The available domain is represented with a set of *scalloped sectors* for efficiency. The work of Jones [16] exploits Voronoi diagrams to improve the generation of Poisson-disk distributions through Lloyd's relaxation and also has $O(N \log N)$ complexity. In 2007 White et al. [8] proposed a hierarchical approach named Hierarchical Dart Throwing (HDT) based on a quadtree subdivision of the sampling domain. In the following we will analyze in depth this algorithm since our Poisson-disk solution for meshes takes inspiration from this approach. A parallel algorithm for the generation of this kind of sampling designed to be efficiently implemented in GPU has been recently proposed by Wei et al. [17]. This algorithm is able to generate samples in n dimensions at the cost of higher computational time.

2.2 Poisson-Disk Sampling on Mesh Domain

The first work for the direct sampling of triangular meshes is the Dual-tiling scheme of Li et al. [1]. Despite the similar name this tiling approach is different to the one proposed by Lagae et al. [13]. This algorithm is based on Wang Tiling and it requires a parameterization of the mesh. The main idea is to build a *dual surface* on the mesh (hence the name of the method) in order to limit the possible kinds of tiles and to simplify the construction of the tile set used for the tiling. A dual surface is such that every vertex is shared exactly by four tiles. The Poisson-disk samples are generated on the tiles and then such tiles are applied to the dual surfaces. The isotropy of the samples generated in this way can be improved at the end of the process by applying Lloyd's relaxation. The authors stated that 10-20 iterations are sufficient to obtain a distribution with good blue noise characteristics.

Fu et al. [6] extended the algorithm of Humphreys et al. [15] to meshes. The geodesic distance is used to find the available boundary sampling. More specifically, they compute geodesic equidistant curves over the mesh surface and use them to handle the boundary where new samples should be placed. They also employed this algorithm for remeshing; after sampling directly the surface a slight variant of the algorithm of Turk [18] is used to retile the mesh. The sharp features are preserved separately by extracting them with the algorithm of Jiao and Heath [19] before the surface sampling. Also in this case some iterations of the Lloyd's relaxation are used to improve the final point distribution.

Cline et al. [3] generalize the HDT by White et al. [8] to generic surfaces including triangular meshes, Bézier patches and implicit surfaces. Since our algorithm is also based on an extension of the approach of White et al. we provide a detailed description and comparative discussion about it in Section 4.

Finally, another work based on the same main idea as ours is the one proposed by Bowers et al. [7]. This algorithm can be considered as concurrent to ours. In order to assess well the scientific contribution of our technique with respect to the algorithm of Bowers et al. we discuss it in a specific Section (5).

```
FOREACH_TRIANGLE t_i
{
  // Poisson(.) is a function which returns an
  // integer random value with a probability
  // that follows a Poisson distribution
  SAMPLEPERTRI = POISSON( n_s · (A(t_i)/A_m) ) ;
  REPEAT SAMPLEPERTRI TIMES
    COMPUTEUNIFORMSAMPLE(t_i);
}
```

Fig. 1. Per-Triangle Monte Carlo sampling using Poisson distribution. This algorithm relies on a robust implementation of $Poisson(.)$.

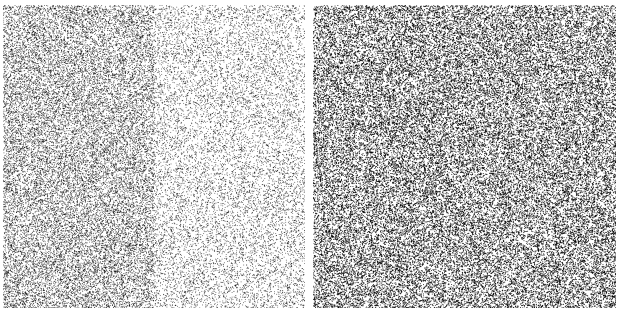

Fig. 2. (Left) Per-Triangle Monte Carlo sampling done using a method for sampling a Poisson distribution designed for small means. (Right) The result when using a method appropriate also for large means.

```
/* Streaming Monte Carlo Sampling (version 1) */
VECTOR<FLOAT> INTERVALS(n_t + 1).
FLOAT SUM =0;
FOREACH TRIANGLE t_i
{
    INTERVALS[I] = SUM;
    SUM += A(t_i);
}
FOR I = 0 TO n_s
{
  VAL = UNIFORMRANDOM ( 0 ... SUM );
  INDEX = FIND(INTERVALS, VAL);
  COMPUTEUNIFORMSAMPLE(t_INDEX);
}
```

```
/* Streaming Monte Carlo Sampling (version 2) */
GENERATE_SINGLE_SAMPLE
{
    // AreaMax = max_i A(t_i)
    WHILE(TRUE)
    {
        INDEX = UNIFORMRANDOM ( 0 ... n_t );
        VAL = UNIFORMRANDOM ( 0 ... 1);
        IF(VAL > A(t_INDEX)/AREAMAX)
        {
            COMPUTEUNIFORMSAMPLE(t_index);
            RETURN;
        }
    }
}
```

Fig. 3. Two versions of the Streaming Monte Carlo sampling algorithm. Version 1 shows the basic idea of bijectively mapping triangle areas onto the real segment, uniformly sampling this segment and getting points into the corresponding triangles. The bottom code fragment (version 2) shows the core procedure for getting rid of the *log* search in the algorithm above. Assuming the ratios of the triangle areas are bounded, the modulo search can be used to jump to a specific triangle and discard it or not in a probabilistic way.

3 MONTE CARLO SAMPLING

Monte Carlo sampling is one of the most classic sampling scheme employed in several problems such as evaluation of integrals, physical simulation, optimization and so on. While some of the considerations here reported could be considered trivial from a statistic point of view, we found it useful to report them because we did not find any specific treatise about the issues here discussed. In particular, the implementation of an algorithm to compute a set of points on a surface according to an uniform distribution requires a bit of care to avoid *biased* Monte Carlo distributions.

In the following, we indicate with n_s the number of samples to generate, with n_t the number of triangles of the mesh, with $A(t_i)$ the area of triangle t_i, and with $A_m = \sum A(t_i)$ the area of the whole mesh. We can model the number of samples which fall into a given triangle t as a random variable with a Poisson distribution with mean $n_s \frac{A(t)}{A_m}$, as shown by the algorithm in Figure 1. This approach has a few possible drawbacks.

Note that we require an *accurate* method for sampling the Poisson distribution (i.e., one that works for both small and large means). Knuth's method [20] is a popular choice but, in this specific case, it is not appropriate, since it is designed for small means. However, there are plenty of methods (including [21]) that are able to accurately sample a Poisson distribution for both large and small means. Figure 2 shows the bias generated when the Poisson distribution is not sampled appropriately. The mesh sampled is one of the stress meshes used in our experiments (see Figure 12).

The second issue is related to the fact that such an approach does not guarantee that the *exact* number of samples n_s is generated; therefore applications that require exactly n_s samples have to perform a second pass over the generated set of samples to prune them or to add further samples. The last issue is that, because of this two-pass approach, this technique cannot easily generate a continuous stream of uniformly random points over the whole mesh, but the samples are generated in a triangle by triangle manner.

To override these problems we can follow a different strategy based on the idea of just simulating the actual sampling process by unfolding all the triangle areas over consecutive segments of the real line, uniformly sampling that portion of the real line and finding to what triangle each sample belongs (Figure 3 (Top)). From a computational complexity viewpoint, searching the triangle corresponding to a given point onto the line has a logarithmic cost, but it is interesting to note that it is possible to write probabilistic approaches with expected constant per-sample complexity. In fact, if all the triangles would have the same size the search for the triangle containing a sample could be resolved in

constant time with a modulo search operation. Moreover, even if the triangles are not equiareal, but a bound like $\frac{1}{2}a < A(t) < a$ is valid, we can expect constant time search by performing a modulo search and then eventually discard samples according to the actual size of the chosen triangle. More precisely, in order to comply with the varying triangle size, we discard triangles with a probability proportional to the size of the approximation (see Figure 3 (Bottom)). For example, if the smallest triangle is just one half of the largest one (e.g. we map each triangle into a fixed size interval that is at most twice the correct length), we can expect that, on the average, the procedure GENERATE_SINGLE_SAMPLE runs the inner loop less than two times.

For the general case, in which we cannot do any restrictive assumption on the triangle areas, we can resort to logarithmic binning: we partition all the triangles according to their size into separate bins of exponentially growing size. In this case the first step is to choose one of the bins with a probability proportional to the sum of the areas of the contained triangles. For any practical instance the number of bins can be reasonably bounded by a small constant, hence we can expect a constant time for getting a single sample.

Summarizing, if we do not need an exact number of samples, the triangle by triangle approach of Figure 1 is a simple solution, assuming a correct choice of the implementation of the random number generation *Poisson(.)*. If we need a continuous stream or an exact number of samples to generate, the two algorithms of Figure 3 provide a still efficient and easy to implement solution.

4 Poisson-Disk Sampling

The two algorithms we propose take inspiration from the same 2D method, the Hierarchical Dart Throwing (HDT) by White et al. [8]. This algorithm is particularly interesting for two reasons. First, its computational complexity is very low; the authors argue that the complexity of HDT is $O(N)$ where N is the number of samples. This statement is not formally demonstrated but statistical considerations based on their experimental results are consistent with it. The second reason is that it can be extended to the domain of polygonal meshes without too much effort. In fact, this approach has just been extended in a recent work of Cline et al. [3] to deal with different types of surfaces such as meshes, Bézier patches and implicit surfaces. Before describing the proposed algorithms we recall the HDT approach and the method of Cline et al. in order to make more clear our contribution and underline the main differences of our approach.

4.1 HDT and its 3D Extensions

Like other recent algorithms, the idea of White et al. [8] is to reduce the 2D sampling domain during the sample generation in order to make the insertion of new samples more efficient. To achieve this a quadtree subdivision of the (planar) sampling domain is used. Initially, the sampling domain (a square of edge length w) is subdivided in cells of size of $\frac{r}{\sqrt{2}}$, where r is the disk radius of the distribution. In this way, approximately $1/4$ of the cells should contain a sample at the end. These cells are added to an *active list* of cells. Then, at each step a cell c is extracted from the active list with a probability proportional to its area, and a new sample p with uniform probability is generated within cell c. If the new sample p violates the radius constraint for some other points the cell is further subdivided and four subcells are added to the active list. Otherwise, the cell is discarded. In order to optimize the number of cells in the active list, only the cells that are not completely covered by current samples are added to the list. Experimental results confirm that at each step the number of active cells reduces exponentially, motivating the conjecture of a linear complexity.

Cline et al. [3] extended the aforementioned algorithm by generalizing the cell subdivision of the sampling domain with a subdivision of the surface into *surface fragments*. Their algorithm makes the same steps: a fragment f is extracted from the active list of fragments, a new sample p is generated inside it, and if p violates the radius constraint, a rule of subdivision is used to subdivided it into sub-fragments. The sub-fragments are added to the active list. Cline et al. proposed different fragments, rules of subdivision, and fast indexing, for different types of surfaces included triangular meshes. The basic fragment for a triangular mesh is the triangle itself. The triangles of the mesh are indexed by a logarithmic binning based on their area.

In the next section, we propose a simpler way to extend the HDT approach. The two main ideas of our approach are the following:

- We do not rely on elements which depend on the mesh but we directly subdivide the 3D space in order to make the algorithm independent of the mesh complexity and topological correctness.
- We generate the samples in a very non-standard way, i.e. we pre-generate a suitable set of samples on the mesh surface (*sample pool*) and we build the Poisson-disk distribution by *removing* samples from the pre-generated pool. We call this approach *Constrained Poisson-disk* sampling since the initial sample pool drives the final sampling distribution.

In the experimental results we will show that this approach is able to efficiently produce Poisson-disk distribution with good properties. Following this approach we propose two algorithms; the *Constrained Hierarchical Cell-based Poisson-disk* and the *Constrained Sample-based Poisson-disk*. These algorithms differ basically in how they manage the sample pool during the sample removal phase.

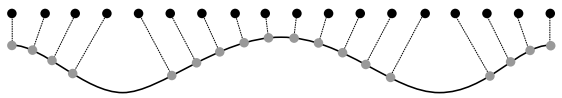

Fig. 4. The closest-point projection operation significantly affects the distribution of the samples. A set of uniform samples close to the surface is no more uniform once projected.

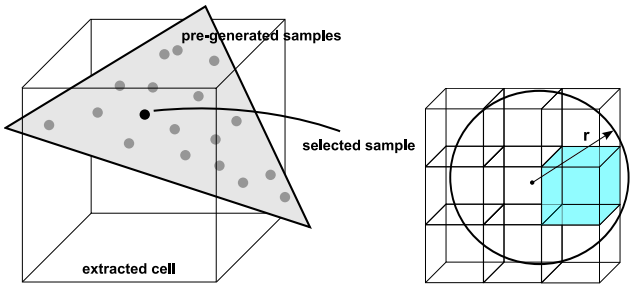

Fig. 5. Sample generation. (Left) Samples are chosen from a pre-generated uniform sampling of the surface. (Right) Samples violating the minimum distance constraint are removed, eliminating all the samples of cells within a sphere of radius r.

4.2 Constrained Hierarchical Cell-Based Poisson-Disk Sampling

The main differences between our approach and the one of Cline et al. concerns the fragment selection and the generation of samples. The use of the mesh triangles as base fragments of the surfaces makes the Cline et al. algorithm dependent on the size and on the connectivity of the mesh to sample. Since we want to make the algorithm independent of the mesh complexity and mesh quality (i.e. triangles' shape) we use a space-subdivision to update the available sampling domain. The 3D space surrounding the mesh is subdivided in cubic cells.

Assuming this, the problem is the generation of a sample on the surface inside each cubic cell. Different solutions are possible to achieve this. A first one is to generate a sample inside the cube with uniform probability, and then to project it on the mesh, discarding the sample if the radius constraint is violated. This way of proceeding may cause two problems: the blue noise characterization could not be respected due to the projection operation, and with high probability the distribution of the final samples would be biased by the geometric features of the mesh (see Figure 4). To avoid such problems we follow an inverse approach: we pre-generate uniformly many samples on the mesh surface, building a *sample pool* ($\mathcal{S}$), and we select randomly one of this pre-generate samples in the cell (see Figure 5). In this way, by construction, the sample lies on the mesh surface. After a sample is selected all the samples of cells within a sphere of radius r are removed from $\mathcal{S}$ in order to optimize the check of violation of the radius constraint. In order to pre-generate an uniform distribution of points on the mesh surface we use the previously discussed Monte Carlo unbiased algorithm (the implemented one is the version 2 of Figure 3). An oversampling factor $o_V \in \mathbb{N}$ controls the number of samples in the sample pool, in other words if N samples have to be generated, No_V samples are distributed over the mesh surface. Note that only if the Monte Carlo distribution is perfectly uniform from a statistical point of view this approach produces a Poisson-disk sampling on the triangular mesh with good properties in terms of blue noise and sample packing. In the Experimental Results Section we will show that the quality of the distribution so generated is high even if no other additional relaxation steps, such as Lloyd's iterations, are employed. This is a very important characteristic since many methods for Poisson-disk sampling need to apply a few Lloyd's relaxation iterations to improve the final quality of the distribution generated. While this is usually not so computationally expensive, it requires strong assumptions on the quality of the underlying mesh. We would like to remark that our approach, once the sample pool has been generated, is *mesh-less* and for that reason it can be employed also for other purposes, for example to process point clouds as shown in Section 4.6.

The pseudo-code of the algorithm is summarized in Figure 6. The EXTRACTCELL(.) procedure chooses randomly one of the non-empty cells of the hierarchy with a probability proportional to the number of pre-generated samples inside it. The ISVALID(.) function controls that the selected sample does not violate the disk radius constraint. From an implementation point of view the hierarchy is handled using a spatial hash method [22] to augment the efficiency of the search of the cell a sample belongs to. The cell size is chosen such that 8 cells are contained in the sphere of radius r in an analogous way of White et al. Anyway, our tests demonstrate that slightly different choices of the cell size do not influence too much the overall performances. An example of sampling distribution generated with this method is shown in Figure 8.

4.3 Constrained Sample-Based Poisson-Disk Sampling

This algorithm is a variant of the constrained cell-based Poisson-disk just described. In practice, the idea is to remove the need of a hierarchical space subdivision and to work only with samples. Every time a sample is generated by extracting it from $\mathcal{S}$, all the samples in a radius of r are removed from the sample pool. This prevents, by construction, any possible violation of the minimum radius constraint. The samples are indexed by a spatial hashing to improve the performances of the sample removal operations. The algorithm ends when no more samples are available in the sample pool. The pseudo-code is given in Figure 7.

The initial shuffle of the samples, which guarantees uniform probability for the sample extraction, is made by

```
PoissonDiskSampling(int N)
{
  // 1. Pre-generate samples on the mesh
  SamplePool pool = GenerateSamplePool(N);

  // 2. Fill a spatial index for fast access to samples
  SpatialHashTable cells = FillSpatialHashTable(pool);

  // 3. Random shuffle of the cells
  randomShuffle(cells);

  // 4. Main Loop
  Samples samples;
  while(cells.isNotEmpty())
  {
    // choose a cell with a probability proportional
    // to the number of samples contained in it
    Cell cell = extractCell(cells);
    // generate a valid sample inside the current cell
    // by extracting it from the pre-computed sample pool
    Sample p = extractFromSamplePool(cell, pool);
    samples.add(p);

    // subdivide cell if necessary and update active cells
    if (isValid(p))
        removeCell(cell, cells);
    else
        subdivideCell(cell, cells);
  }

  return samples;
}
```

Fig. 6. Constrained Hierarchical Cell-based Poisson-disk sampling.

```
PoissonDiskSampling(int N, float radius)
{
  // 1. Pre-generate samples on the mesh
  SamplePool pool = GenerateSamplePool(N);

  // 2. Fill a spatial index for fast access to samples
  SpatialHashTable cells = FillSpatialHashTable(pool);

  // 3. Main loop
  Samples samples;
  while(pool.isNotEmpty())
  {
    // generate a valid sample inside the current cell
    // by extracting it from the sample pool
    Sample p = extractFromSamplePool(pool);
    samples.add(p);

    // remove samples in the disk-radius
    removeSamples(p, radius, cells, pool);
  }

  return samples;
}
```

Fig. 7. Constrained Sample-based Poisson-disk Sampling

exploiting cell partitions like in the previous algorithm. Any other spatial indexing structure could be used to efficiently remove from $\mathcal{S}$ the invalid samples (the ones within distance r), but the use of hashed grid make easy the initial shuffling process.

4.4 Euclidean vs Geodesic Distance

In the context of Poisson-disk sampling of meshes different metrics for computing distances between samples can be used. Until now we have always intended the Euclidean distance, but the geodesic one is another

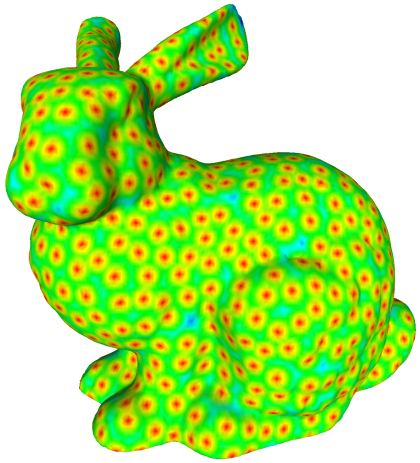

Fig. 8. Poisson-disk sampling of the Bunny model. The inter-distance between samples is visualized through color gradient. This helps to visualize the selected disk radius for each sample and the sample packing.

sound metric. The Geodesic distance can be considered more suitable because it better reflects the features of the sampling domain (the mesh surface), conversely the Euclidean distance can cause unwanted effects: samples on the two sides of thin surfaces can conflict. On the other hand the Euclidean distance is by far simpler to be computed, more efficient and much more robust w.r.t. the underlying mesh. Moreover, the Euclidean distance allows to solve the sampling problem with a meshless approach that widens the application domain to point clouds. Another possibility, available in our provided implementation, is to use an approximation of the geodesic distance, like the one of [7], that needs only position and normal of the involved samples. This approximation is sufficient in many cases to avoid most of the problems caused by the use of the plain Euclidean distance.

4.5 Importance Sampling

Both the techniques here described can be modified in a natural way to obtain *importance sampling*. In practice, it is sufficient to assign for each sample of the sample pool a different disk radius r, according to the criterion used to determine the "importance". The value of the disk radius should be inversely correlated with the importance; in this way the important parts of the mesh will have a sample density higher than the parts with low importance. More control to the distribution generated in this way can be achieved by assigning a minimum and a maximum density factors, associated respectively to the minimum and the maximum of the importance values (v_{inf} and v_{sup}). In formula:

$$r(v) = \frac{rk - r/k}{v_{sup} - v_{inf}}(v - v_{inf}) + \frac{r}{k} \quad , \quad k > 1 \qquad (1)$$

where v is the value of the importance of a certain sample, $r(v)$ its disk radius, and k controls both the density factors, the one associated with the maximum

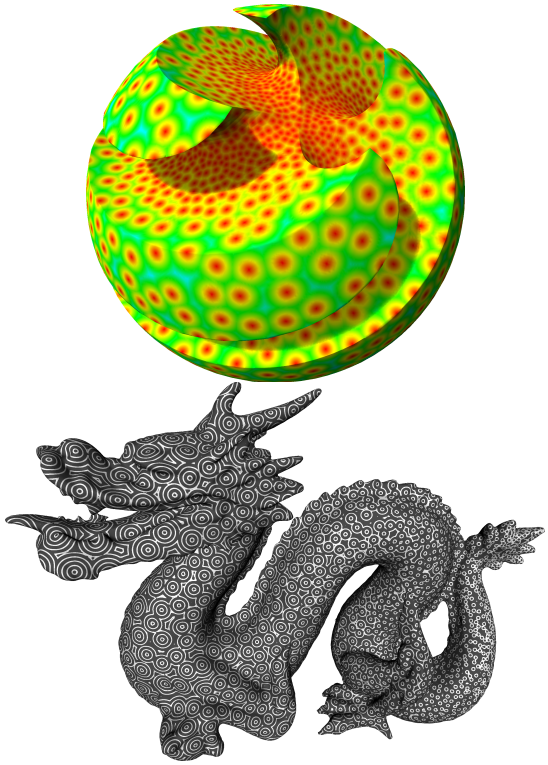

Fig. 9. Examples of importance sampling. (Top) The model is sampled with an importance function proportional to the distance from sphere center. (Bottom) The Dragon model is sampled using the importance function $f(x, y, z) = x$. In this case the contour of the radius is visualized with alternating stripes.

value (rk), and the one associated with the minimum value (r/k). Figure 9 shown two examples of importance sampling.

4.6 Playing with the Sample Pool

One of the distinctive characteristic of the two proposed Poisson-disk algorithms is that, at the end, the final set of samples is a subset of a given one. While an initial uniform distribution of samples is needed for getting a sound Poisson-disk distribution, as shown in Section 6, it is important to highlight that we can exploit this property in a variety of ways to tweak the distribution generation according to the specific application's needs. Here, we present some applications to demonstrate the flexibility of our method.

A first example is to use the proposed method for subsampling point clouds, since the meshless nature of our approach makes the mesh surface unnecessary for the process of choosing a number of samples complying with the Poisson empty disk property. See Figure 11 for an example of such a subsampling. Note how the Monte Carlo subsampling of the original point cloud (second figure from the left) preserves the original density differences resulting in a non-uniform sampling density of the subsampled point cloud.

Another use of our approach is to constrain the sampling picking process in a general way. For example, in Figure 10 a Poisson-disk distribution is generated such that it is guaranteed that feature edges are sampled in a "preferred" way. The sample pool strategy allows to tweak the sampling process as follows: start by random sampling the edges of the mesh, then runs one of the two proposed algorithms. At this point you have a sample distribution over the surface that respect the Poisson empty disk property but it is by far non-maximal. The next step is simply to restart the sampling process using as first points the one previously chosen (e.g. use them to prune out unnecessary points from the sample pool). It is possible to notice from the Figure 10 how the crease edges have been prioritized in the sampling process generating a Poisson-disk distribution with as much as possible samples on the edges. A sample distribution of this kind, which preserve the mesh features, can directly be used for high quality remeshing purposes following, for example, the approach described in Fu and Zhou [6].

5 Constrained vs Parallel Poisson-Disk Sampling

While our approach takes inspiration from the HDT of Cline et al., the algorithm of Bowers et al. [7] (we refer to it with PPD in the following) is an extension of the work of Li-Yi Wei [17] for the parallel generation of N-dimensional Poisson-disk samples using the GPU. The main idea is to regularly subdivide the sampling domain and to arrange the space elements in subsets (called *phase groups*) which can be processed independently. In this way the sample generation can be parallelized efficiently. The extension of this algorithm to surfaces consists in subdividing the bounding box of the 3D object into grid cells arranged in phase groups, and then generating the samples on the surface by extracting them from a pre-generated set, exactly as in our approach. The main advantage of this algorithm with respect to our is the parallelization of the algorithm (implemented in GPU) thanks to the phase group subdivision of the grid cells. Another important feature of the PPD algorithm is that it works also considering an approximate geodesic distance between samples and not only Euclidean distance. This geodesic approximation can be used in our approach as well (and infact the provided reference implementation supports it), but for a simpler discussion in this paper we discuss just the case of the classic 3D euclidean distance.

Another difference between the PPD and our algorithms, regards the strategy to handle conflicts between generated samples. In the PPD, each new sample is checked against neighborhood samples to ensure that the radius constraint is respected. In our case, for the cell-based algorithm, the conflicts to check are greatly reduced since we remove every cell within the sphere of radius r; for the sample-based algorithm, we remove *every* sample inside the sphere of radius r for each

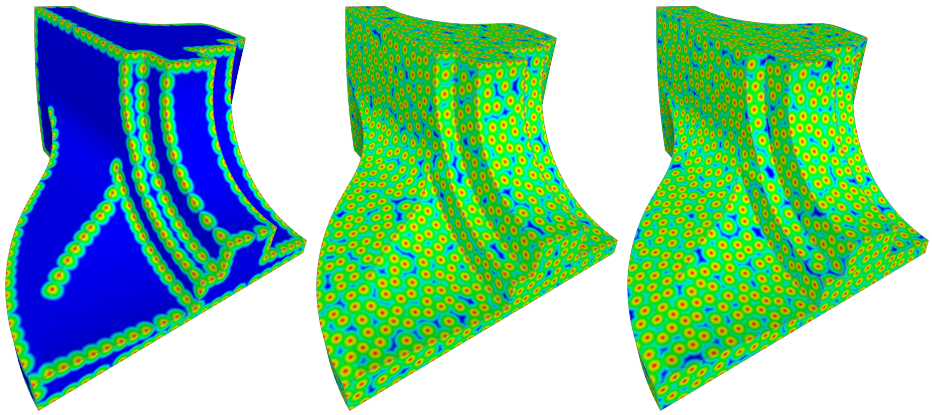

Fig. 10. Constrained Poisson-disk sampling can be used for generating very specific point distributions. We start (Left) by using a sample pool containing points lying only onto the mesh sharp edges. In a second step (Middle) when we are unable to add other points from this pool, we switch to a sample pool containing points uniformly sampled from the whole surface. (Right) For comparison, a standard Poisson-disk sampling build using a uniform Monte Carlo sample pool, where the probability that a sample falls exactly over the edges is zero.

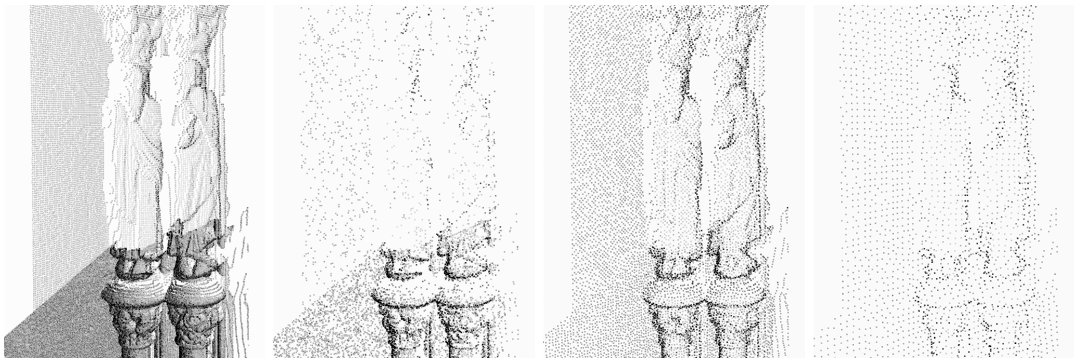

Fig. 11. Our Poisson-disk sampling algorithm can be used to effectively subsample in a fair way a non uniform dense point cloud. (From Left to Right) The original point cloud (a detail), a Monte Carlo subsampling, two different Constrained Poisson-disk subsampling (with different disk radius).

new sample we add; hence, by construction, we do not have to handle conflicts. This advantage is paid by introducing a dependence on the oversampling factor on the computational time. Despite this problem, our implementation is efficient since the query to retrieve the nearest samples (using the spatial hash table as explained) introduces only a small overhead. Even if more efficient from a computational time viewpoint, the "dart throwing" strategy of the PPD requires a fixed number of trials to bound the computational time. So, theoretically, there is the possibility to leave some grid cells without a sample, compromising the quality of the sample packing. The probability that this happens increases with the percentage of the cubic cell covered by previously generated samples. We underline that this probability is very low (e.g. around 0.1% assuming 6 trials and a cell covered at 60% which contains 20 samples), it should be noted that this problem is completely avoided in our sample-based algorithm. This potential missing of samples can worsen when the sample pool is not generated in a random-uniform way like in the example of geometrically-constrained blue noise distribution shown in Figure 10.

Concerning importance sampling, our techniques can be naturally extended by assigning a different r for each sample. The same can be done also for the PPD algorithm but this complicates the management of conflicts causing a significant slowdown in the performance of the PPD algorithm passing from a maximum of 200,000 samples/sec for complex models to 20,000 samples/sec (according to the results presented in the paper by the authors). In this case the speed is comparable with the sample-based version of our algorithm, as shown in the Experimental Results Section.

Concluding, we can state that the PPD algorithm outperforms the ours thanks to its parallel implementation, but our algorithms are more flexible and allow easy tweaking for many purposes, such as importance sampling or geometrically-constrained blue noise sampling. Moreover, the sample-based version is trivial to implement.

6 Result Analysis

Here we present some experimental results to demonstrate that the proposed algorithms are efficient and able to generate samples distribution with good properties. First we present, a frequency analysis to show that the two Constrained Poisson-disk Sampling algorithms have blue noise characterization. Then we evaluate the sample packing, i.e. the quality of the sampling, using the relative radius [2]. Finally, the computational time of the proposed algorithms are presented.

6.1 Frequency Characterization

Typically, the Poisson-disk sampling algorithms are evaluated using two mathematical tools, i.e. the *periodogram* ($\mathcal{P}(f)$) and the *relative radius* (ρ) [2].

The periodogram is defined as the Fourier Transform of the auto-correlation function of a signal. For a real signal this corresponds to the square of the magnitude of the Fourier Transform:

$$\mathcal{P}(f) = \mathcal{F}(\mathcal{A}(f(x)) = \|\mathcal{F}(f(x))\|^2 \quad (2)$$

where $\mathcal{A}(f)$ is the auto-correlation function of f. By assuming to replace each sample of a given distribution with a delta of Dirac function the distribution can be rigorously characterized by its periodogram. For example, an uniform distribution, like the one produced by a Monte Carlo sampling algorithm, should exhibits a periodogram close to white noise since there are no specific pattern at any frequency that emerge. The periodogram of a Poisson-disk distribution, instead, is close to a blue noise spectrum. This effect is caused by the minimum distance constraint between each samples.

This analysis can be easily computed for a planar domain by applying the 2D Discrete Fourier Transform (DFT) to the samples but it is not extendable to a mesh domain. One possible solution to overcome this problem could be to parameterize the surface and then to apply this kind of frequency analysis on the parameterized samples. Unfortunately, in this way, the parameterization itself introduces distortions in the periodogram. This could make the analysis not completely clear and reliable. This problem can be alleviated by sampling developable surfaces. Here we opt for an approach often used in the Computer Graphics community when mesh sampling is treated, which consists in generating samples on a planar surface and then compute the periodogram on this surface. In order to stress our algorithms we employ planar meshes with an highly irregular connectivity and density. These *stress meshes* are shown in Figure 12. Three meshes are used: a quad-grid, with equi-spaced vertices and regular connectivity, another grid with decreasing space between its vertices and almost regular connectivity, and a non-uniform grid with highly irregular connectivity and density. The periodograms of these meshes sampled with our algorithms is evaluated. The results obtained are shown in the next figures. The red graphs correspond to the *average radial power* ($R_p(f)$) of the periodogram while the blue graph is the *average radial anisotropy* ($R_a(f)$). The average radial power is the power of the signal computed by averaging the periodogram on an anuli of a given radius (Δf). This is useful to characterize the spectrum of the distribution. In formula:

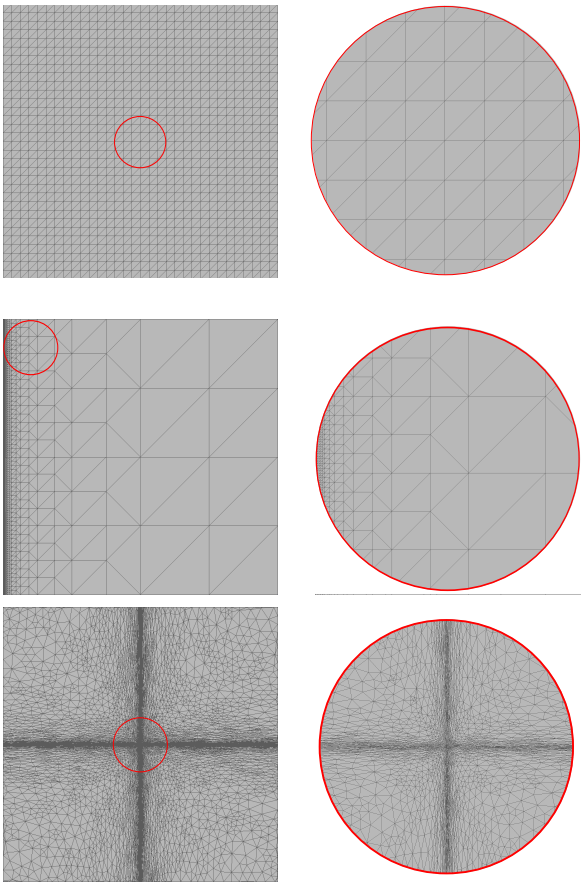

Fig. 12. The stress meshes used in the sampling tests.

$$R_p(f) = \frac{1}{2\pi f \Delta f} \int_f^{f+\Delta f} \int_0^{2\pi} \mathcal{P}(f cos\theta, f sin\theta) d\theta df \quad (3)$$

The average radial anisotropy is obtained considering the variance of the power of the signal on an anuli of given radius.

$$R_a(f) = \frac{1}{2\pi f \Delta f} \int_f^{f+\Delta f} \int_0^{2\pi} \left(\mathcal{P}(f cos\theta, f sin\theta) - \bar{P}\right)^2 d\theta df \quad (4)$$

where $\bar{P}$ is the mean of the power spectrum calculated on the anuli of interest. This is a measure of the radial symmetry of the power spectrum and it is useful to pose in evidence if there are preferred "directions" in the spectrum, i.e. some bias or emerging pattern. The periodogram and the graphs here presented are generated using the Point Set Analysis (PSA) tool [23], a software tool to evaluate frequency characteristic of sampling patterns.

The frequency characterization of the Poisson-disk sampling is particularly important. As possible to see

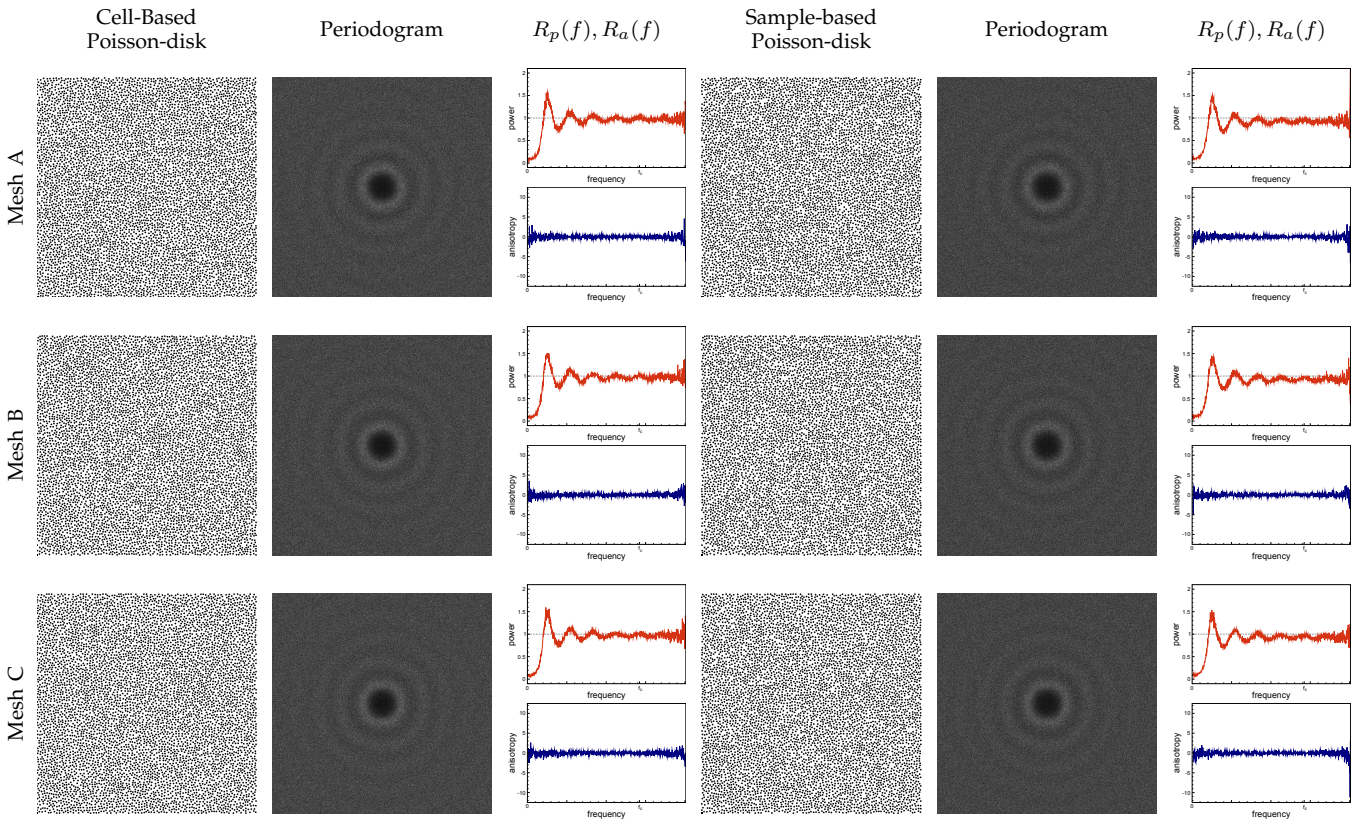

Fig. 13. Frequency analysis of the two versions of the Constrained Poisson-disk sampling proposed. Radial Power $(R_a(f))$ is shown in red and Radial Anisotropy $(R_a(f))$ in blue. About 5,000 samples are generated for each mesh.

in Figure 13 both the hierarchical cell-based and the sample-based algorithm are able to produce distributions with good blue noise properties. Another important issue to point out is that the frequency properties are independent from the mesh connectivity as expected by the design of our algorithms. Obviously, the sample pool generation influences a lot the frequency properties of the final distribution; Figure 15 depicts the behavior of the periodogram varying the oversampling factor o_V of the pre-generated samples. The blue noise pattern emerges while the Monte Carlo oversampling increases. This happens because the higher is the oversampling factor, the higher is the approximation of the initial distribution in the sample pool of a perfectly uniform distribution. We can note that even a small oversampling factor is able to exhibit good blue noise properties, this is an important aspect that makes the proposed techniques particularly efficient in generating an approximate Poisson-disk distribution very quickly. We can conclude that by choosing an appropriate distribution to initialize the sample pool we are able to generate samples distribution with the desired frequency properties.

6.2 Radius Statistics

Concerning the quality of the sample packing of the Poisson disk distribution generated we calculate, on the stress mesh, the *relative radius* (ρ), i.e. a fraction of the disk radius r defined as:

$$\rho = \frac{r}{r_{\max}} \tag{5}$$

where $r_{\max}$ is the radius to obtain the maximum packing density of N disks distributed on a toroidal domain (see [2] for further details). The relative radius has the range between 0.65 and 0.85 to achieve a "good" packing avoiding regular configurations. For good packing we intend that almost all the available sampling domain has received an uniform number of samples. ρ has been calculated for each stress mesh averaging on different number of samples. Both the cell-based and the sample-based algorithm have been tested obtaining similar results. In summary, the ranges of values of ρ obtained is between 0.595 and 0.695, but in the majority of the cases it ranges between **0.64** and **0.68**. Considering that no further processing is done to improve the sample packing quality, e.g. Lloyd's relaxation, we can state that our algorithms are able to generate sample distributions with high quality.

6.3 Performances

As just stated, one of the objectives of this work is to design efficient algorithms for triangular mesh sampling.

Unbiased Monte Carlo Sampling			
Model	Samples	Time	Samples/s
FanDisk	1,000,000	718	1,392,757
Bunny	1,000,000	837	1,194,743
Dragon	1,000,000	1,391	718,907
FanDisk	10,000,000	7,171	1,394,505
Bunny	10,000,000	9,235	1,082,837
Dragon	10,000,000	13,297	752,049

Poisson-disk Sampling					
Model	Type	o_V	Samples	Time	Samples/s
FanDisk	Cell-bas.	$\times 20$	125,794	12,438	10,113
Bunny	Cell-bas.	$\times 20$	126,664	14,438	8,772
Dragon	Cell-bas.	$\times 20$	126,421	15,031	8,410
FanDisk	Cell-bas.	$\times 10$	117,628	6,640	17,715
Bunny	Cell-bas.	$\times 10$	118,062	7,313	16,144
Dragon	Cell-bas.	$\times 10$	117,839	7,656	15,391
FanDisk	Sample-bas.	$\times 20$	130,939	10,859	12,058
Bunny	Sample-bas.	$\times 20$	131,126	11,407	11,495
Dragon	Sample-bas.	$\times 20$	130,903	12,051	10,862
FanDisk	Sample-bas.	$\times 10$	121,400	5,313	22,849
Bunny	Sample-bas.	$\times 10$	121,632	5,640	21,565
Dragon	Sample-bas.	$\times 10$	121,446	5,922	20,507

Fig. 14. Computational times to sample meshes of different size. The models sampled are the FanDisk ($\approx$ 12,000 faces), the Bunny ($\approx$ 70,000 faces) and the Dragon ($\approx$ 1,000,000 faces). Timings of algorithms are almost independent from the mesh size and quality. Times are in milliseconds.

Hence, the computational time required to generate a certain number of samples is an important factor to evaluate. The proposed algorithms have been tested on three different models in order to assess their respective performances: the FanDisk model ($\approx$ 12,000 triangles), the Bunny model ($\approx$ 70,000 triangles) and the Dragon model ($\approx$ 1,000,000 triangles). The machine used for the tests is an Intel Core Duo 1.86GHz with 2GB RAM. Table 14 summarizes the algorithms' performances.

We can easily notice that the overall performances of the proposed algorithms are, in general, very good and that the mesh complexity and triangle shape do not influence them. The unbiased version of the Monte Carlo distribution is able to generate about 1,000,000 points/sec on any model tested. We recall that the version tested is the one used to generate the sample pool, i.e. the version 2 of Figure 3. The other one (version 1) has similar performance. Both our cell-based and sample-based Poisson-disk sampling algorithms are very fast with respect to other state-of-the-art algorithms which generate Poisson-disk distribution on meshes. In particular, considering the experimental results reported in the corresponding papers, we can state that our algorithms outperform the ones of Li et al. [1] and Fu et al. [6] and, in some cases, is faster than the one of Cline et al [3]. In fact, the algorithm of Cline et al. is able to generate about 20,000 points/sec for models with about 500K faces but since it decrease its performances with the number of triangles of the model we can argue that our method is considerably faster for models with millions of triangles. The algorithm by Li et al. [1] takes more or less 30 seconds to generate 20,000 samples on the Bunny model, hence, our approach is more than one order of magnitude faster. The algorithm by Fu et al. [6] seems even more slow than the one of Li et al. but we have to point out that also the remeshing operation time is included in the experimental results reported in their paper.

7 Concluding Remarks

In this paper we propose a new flexible and efficient Poisson-disk sampling scheme for triangular meshes and provide a discussion about efficient implementation of Monte Carlo sampling for meshes.

The novel approach for Poisson-disk sampling of triangular meshes has been proposed in two variants. These two algorithms follow a new paradigm: the generation of the distribution by removing samples from an initial uniform dense distribution. This paradigm allows the generation of what we call Constrained Poisson-disk distribution, i.e. a sampling scheme that approximates very well a Poisson-disk distribution with blue noise properties, but that can also include other geometric constraints. This approach can be used also for other purposes, such as subsampling of point clouds and remeshing, and can be applied to any type of surface for which an initial, dense and uniform sample pool can be generated. Some aspects of the main idea have been independently proposed also in the concurrent work of Bowers et al. underlying that this idea is intriguing and worthwhile further research.

The experimental results show that the proposed algorithm is able to generate distributions with very good properties in terms of frequency characterization and Poisson-disk sample packing. The performances of the proposed algorithms have been accurately evaluated demonstrating their efficiency. A publicly available implementation is also made available within the open source mesh processing software MeshLab [9].

Acknowledgments

This research work is partly funded by the EU Community's FP7 ICT under the V-MUST.net project (Grant Agreement 270404).

References

[1] H. Li, K.-Y. Lo, M.-K. Leung, and C.-W. Fu, "Dual poisson-disk tiling: An efficient method for distributing features on arbitrary surfaces," *Visualization and Computer Graphics, IEEE Transactions on*, vol. 14, no. 5, pp. 982–998, Sept.-Oct. 2008.

[2] A. Lagae and P. Dutré, "A comparison of methods for generating Poisson disk distributions," *Computer Graphics Forum*, vol. 21, no. 1, pp. 114–129, 2008.

[3] D. Cline, S. Jeschke, K. White, A. Razdan, and P. Wonka, "Dart throwing on surfaces," *Computer Graphics Forum*, vol. 28, no. 4, pp. 1217–1226.

[4] R. L. Cook, "Stochastic sampling in computer graphics," *ACM Trans. Graph.*, vol. 5, no. 1, pp. 51–72, 1986.

[5] M. A. Z. Dippé and E. H. Wold, "Antialiasing through stochastic sampling," in *SIGGRAPH '85: Proceedings of the 12th annual conference on Computer graphics and interactive techniques*. New York, NY, USA: ACM, 1985, pp. 69–78.

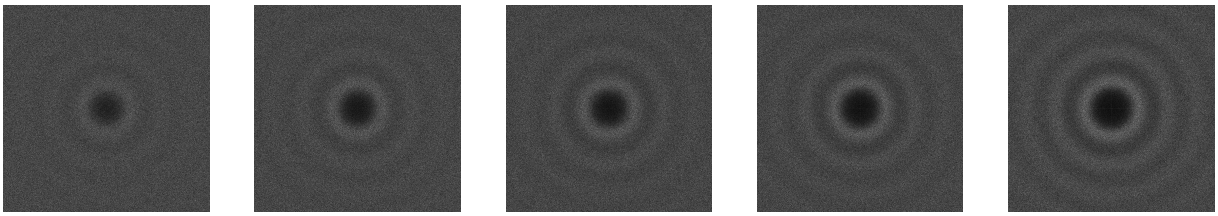

Fig. 15. Frequency analysis of the effect of varying the sample density (o_V) of the pre-generated Monte Carlo samples. As the density increase the Poisson-disk frequency characterization emerges from the white noise. Note that just a lower density could exhibits good blue noise properties.

[6] Y. Fu and B. Zhou, "Direct sampling on surfaces for high quality remeshing," in *SPM '08: Proc. of the Symposium on Solid and physical modeling*. New York, NY, USA: ACM, 2008, pp. 115–124.

[7] J. Bowers, R. Wang, L.-Y. Wei, and D. Maletz, "Parallel Poisson Disk Sampling with Spectrum Analysis on Surfaces," in *ACM SIGGRAPH Asia 2010 papers*, ser. SIGGRAPH ASIA '10. New York, NY, USA: ACM, 2010, pp. 166:1–166:10.

[8] K. White, D. Cline, and P. Egbert, "Poisson disk point sets by hierarchical dart throwing," *Interactive Ray Tracing, 2007. RT '07. IEEE Symposium on*, pp. 129–132, Sept. 2007.

[9] P. Cignoni, M. Corsini, and G. Ranzuglia, "Meshlab: an open-source 3d mesh processing system," *ERCIM News*, vol. 73, pp. 45–46, 2008. [Online]. Available: http://meshlab.sourceforge.net

[10] M. F. Cohen, J. Shade, S. Hiller, and O. Deussen, "Wang tiles for image and texture generation," in *SIGGRAPH '03: ACM SIGGRAPH 2003 Papers*. New York, NY, USA: ACM, 2003, pp. 287–294.

[11] J. Kopf, D. Cohen-Or, O. Deussen, and D. Lischinski, "Recursive wang tiles for real-time blue noise," *ACM Transactions on Graphics (Proceedings of SIGGRAPH 2006)*, vol. 25, no. 3, pp. 509–518, 2006.

[12] A. Lagae and P. Dutré, "A procedural object distribution function," *ACM Trans. Graph.*, vol. 24, no. 4, pp. 1442–1461, 2005.

[13] A. Lagae and P. Dutré, "Poisson sphere distributions," in *Vision, Modeling, and Visualization 2006*, L. Kobbelt, T. Kuhlen, T. Aach, and R. Westermann, Eds. Berlin: Akademische Verlagsgesellschaft Aka GmbH, November 2006, pp. 373–379.

[14] V. Ostromoukhov, C. Donohue, and P.-M. Jodoin, "Fast hierarchical importance sampling with blue noise properties," *ACM Trans. Graph.*, vol. 23, no. 3, pp. 488–495, 2004.

[15] D. Dunbar and G. Humphreys, "A spatial data structure for fast poisson-disk sample generation," *ACM Trans. Graph.*, vol. 25, no. 3, pp. 503–508, 2006.

[16] T. R. Jones, "Efficient generation of poisson-disk sampling patterns," *Journal of Graphics Tools*, vol. 11, no. 2, pp. 27–36, 2006.

[17] L.-Y. Wei, "Parallel poisson disk sampling," *ACM Trans. Graph.*, vol. 27, no. 3, pp. 1–9, 2008.

[18] G. Turk, "Re-tiling polygonal surfaces," *SIGGRAPH Comput. Graph.*, vol. 26, no. 2, pp. 55–64, 1992.

[19] X. Jiao and M. T. Heath, "Feature detection for surface meshes," in *Proceedings of 8th International Conference on Numerical Grid Generation in Computational Field Simulations*, 2002, pp. 705–714.

[20] D. Knuth, *The art of computer programming: Seminumerical algorithms, volume 2, 3rd ed.* Reading, MA: Addison-Wesley,, 1997.

[21] W. Press, S. Teukolsky, W. Vetterling, and B. Flannery, *Numerical recipes in C*. Cambridge Univ. Press Cambridge, 1992.

[22] M. Teschner, B. Heidelberger, M. Mueller, D. Pomeranets, and M. Gross, "Optimized spatial hashing for collision detection of deformable objects," *Proceedings of Vision, Modeling, Visualization (VMV 2003)*, pp. 47–54, 2003.

[23] T. Schlmer and O. Deussen, "Towards a standardized spectral analysis of point sets with applications in graphics," University of Konstanz, Tech. Rep., May 2010.

Massimiliano Corsini is a Researcher with CNR-ISTI. He received a PhD degree in Information and Telecommunication Engineering at the University of Florence in 2005. His research interests are in the fields of Computer Graphics, Computer Vision and Image Processing and include 3D watermarking, perceptual metrics, image-based visual appearance acquisition and modeling and image relighting.

Paolo Cignoni is a Senior Research Scientist with CNR-ISTI. He received a Ph.D. Degree in Computer Science at the University of Pisa in 1998. He has been awarded "Best Young Researcher" by the EG association in 2004. His research interests cover Computer Graphics fields ranging from visualization and processing of huge 3D datasets, to 3D scanning in the cultural heritage field and to Scientific Visualization. He has published more than one hundred papers in international refereed journals and conferences.

Roberto Scopigno is a Research Director with CNR-ISTI and leads the Visual Computing Lab. He graduated in Computer Science at the University of Pisa in 1984. He is engaged in research projects concerned with 3D scanning, surface reconstruction, multiresolution, scientific visualization, and cultural heritage. He published more than hundred fifty papers in international refereed journals/conferences. Roberto has been responsible person for CNR-ISTI in several EU projects and co-chaired several international conferences. He is member of the Eurographics Association, was awarded the EG "Outstanding Technical Contribution Award" in 2008, served as Chair of the Eurographics Association (2009-2010), Co-Editor in Chief of the Computer Graphics Forum journal (2001-2010) and member of the Editorial Board of the ACM J. on Computing and Cultural Heritage.