CEIG - Spanish Computer Graphics Conference (2017)
F. J. Melero and N. Pelechano (Editors)

Fast Stippling based on Weighted Centroidal Voronoi Diagrams

Eila Gómez[1], Elías Méndez[1], Germán Arroyo[1,2] y Domingo Martín[1,2]

[1]Laboratorio de Realidad Virtual, Universidad de Granada
[2]Departamento de Lenguajes y Sistemas Informáticos, Universidad de Granada

Abstract

Stippling is an artistic technique that has been used profusely in antiquity. One of the main problems is that it requires great skill and patience to achieve excellent results due to the large number of points that must be drawn even for small formats. The use of computers, in general, and GPUs, in particular, with their computing capacity, has allowed to overcome many of these limits. We present a real-time GPU stippling program that combines the advantages of positioning based on Weighted Centroidal Voronoi Diagrams and the realistic aspect of the scanned points.

CCS Concepts

• ***Computer graphics*** → *Non-photorealistic rendering;*

1. Introducción

El punteado es una técnica artística que consiste en producir imágenes compuestas por pequeños puntos. Normalmente se usa una plumilla con tinta negra para dibujar sobre papel blanco. Al igual que otras formas de creación gráficas, en muchos casos ha sido sustituida por la fotografía. Con todo, sigue siendo usada en ciertas áreas de ciencias como la botánica, entomología, arqueología, etc., así como en las técnicas de restauración o en la obtención de tatuajes. El punteado presenta varias ventajas frente a otras técnicas de dibujo, basadas en la simplicidad de la primitiva de dibujado, el punto, el cual no presenta orientación, sólo necesita de una tinta, y permite, mediante la variación de la densidad, representar tanto tono como textura y forma.

El problema de esta técnica es que necesita de la habilidad y experiencia del artista y, sobre todo, de mucho tiempo y esfuerzo para conseguir resultados de calidad incluso en formatos pequeños. Por ejemplo, una imagen tamaño A3 puede necesitar de millones de puntos usando una plumilla de 0.20 mm (véase como ejemplo el trabajo de Miguel Endara en `http://vimeo.com/33091687`). Siendo una técnica artística ha recibido la atención de la comunidad de investigadores en Visualización Expresiva desde hace unos 18 años. En este periodo se ha producido un gran avance desde los primeros métodos de posicionamiento [DHvOS99,Sec02b] hasta los desarrollados en los últimos años con características realistas [KMI*09,MALI10].

En todos los desarrollos que se han realizado hasta la actualidad, los algoritmos que se han propuesto se han centrado en las capacidades de los ordenadores para realizar tareas de bajo nivel eficiente y rápidamente, pero no así para simular las propiedades artísticas de la obra realizada manualmente. Así por ejemplo, el ordenador ha sido capaz de calcular las posiciones de los puntos, pero no se ha conseguido que seleccione correctamente, para que tengan una apariencia manual, los puntos que conforman la imagen. A pesar de estas limitaciones, hemos de decir que los resultados que se obtienen hoy en día son bastante buenos, pudiendo generarse en un tiempo aceptable.

Donde los procedimientos actuales no llegan para la replicación de los procesos mentales capaces de sintetizar la obra, se abre la posibilidad de combinar la potencia del ordenador con la capacidad expresiva y estética de un artista, permitiendo que este último solo se encargue de las tareas de alto nivel, dejando el trabajo pesado y repetitivo al ordenador.

En este trabajo se plantean dos objetivos: el primero es facilitar el dibujado en tiempo real o interactivo para que el usuario pueda tomar decisiones de diseño sobre la marcha y poder obtener el resultado deseado, dejando al ordenador la tarea del posicionamiento y el dibujado de los puntos. El segundo es producir imágenes lo más realistas posible.

Para ello hemos implementado un método de posicionamiento muy usado en trabajos previos de punteado, como son los Diagramas de Voronoi Centroidales Pesados (DVCP, *Weighted Centroidal Voronoi Diagram*), debido a su capacidad para alinear los puntos en ciertas características de la imagen. El problema es que suelen ser lentos, por lo que se han desarrollado versiones más rápidas que trabajan en la GPU. Nuestra solución se basa también en GPU, pero aportando el que se usen nuevas capacidades de las tarjetas gráficas

DOI: 10.2312/ceig.20171214

(OpenGL 4.5), haciéndola más sencilla y fácil de comprender. Además, y para producir resultados realistas, integramos el uso de puntos escaneados, también implementando su visualización mediante la GPU. Para la obtención del realismo, nos hemos decantado por el uso de puntos reales escaneados y ajustados a la densidad del dispositivo de salida. También este proceso se ha implementado en GPU. De esta manera hemos conseguido un sistema que que permite trabajar o bien de forma estática a partir de fotografías o bien de forma interactiva, obteniendo en este último caso frecuencias de refresco que van del tiempo real al tiempo interactivo dependiendo del número de puntos y el tamaño de la imagen.

2. Trabajos previos

Los trabajos sobre punteado son numerosos y admiten gran variedad de categorías. Así por ejemplo, podemos encontrar muchos trabajos relacionados con la creación de distribuciones de puntos, pues el posicionamiento de los puntos es una tarea necesaria del proceso, pero hay que tener en cuenta que el punteado es sólo una posible aplicación de la generación de distribuciones de puntos. También hay trabajos que usan modelos 3D (p.e [LME*02,MS02,MPS04]), modelos escaneados (p.e. [XC04]), modelos implícitos (p.e. [FJW*05, SIJ*07]), realizan estudios estadísticos (p.e. [INC*06]), o que evalúan la bondad de una distribución (p.e. [MIA*08]). En este trabajo queremos centrarnos en aquellos que están más relacionados con la generación de punteado tradicional en sus distintas posibilidades.

Si hacemos un recorrido histórico, debemos comenzar por los trabajos de Deussen et al. [DHvOS99, DHvOS00]. En estos se centra la atención en el posicionamiento de los puntos, utilizando el método de los diagramas de Voronoi en el que los centroides son movidos hasta alcanzar una situación de equilibrio, lo que se conoce como Diagrama de Voronoi Centroidal (DVC), el cual se puede conseguir con el método de Lloyd [Llo82, MF92]. Para evitar ciertas características de regularidad y permitir una mayor expresividad, se permitía el control del posicionamiento mediante distintos pinceles. La distribución inicial se basa en una imagen de medios tonos.

Secord [Sec02b, Sec02a] plantea una importante mejora al "pesar" el posicionamiento mediante los tonos de la imagen de entrada. Así, el tono de los píxeles de cada región de Voronoi es tenido en cuenta al mover los centroides, haciendo que las zonas oscuras de la imagen tengan más peso que las zonas claras, lo cual produce un movimiento de los centroides hacia dichas zonas oscuras, dejando más desocupadas las zonas claras. La posibilidad de usar otras primitivas diferentes de los puntos es tratado en [HHD03] mediante una variante de DVC.

El método de los Renderbots es desarrollado por Schlechtweg et al. [SGS05]. Consiste en usar una especie de sistema conductual en el que cada partícula se ve guiada y controlada por información que es guardada en planos de imagen, como por ejemplo el color, o el valor del gradiente, etc.

Aunque el objetivo del trabajo de Smith et al. [SLK05] es la producción de animación con mosaicos, una de las posibilidades es usar puntos como piezas. En general, los métodos para crear mosaicos se basan en el uso de los DVC y variantes ([Hau01]). El trabajo de Kopf et al. [KCODL06] destaca por la posibilidad de crear puntos indefinidamente lo cual permite hacer un zoom sin limitación. Para ello se utilizan el teselado de Wang (*Wang tiling*). La idea es distribuir puntos en las piezas de un teselado de Wang de tal manera que dos piezas adyacentes no presenten problema en su combinación. Dada esta característica y la propia del teselado, es posible producir un conjunto infinito de baldosas de manera recursiva permitiendo el zoom. Dalal et al. [DKLS06] presenta una variante de los mosaicos que también permite el uso de distintas primitivas y su generación para animación. Usa la transformada de Fourier para comprobar la validez de las distribuciones obtenidas.

La idea de que hay características en la imagen que deben ser reforzadas, como por ejemplo los bordes, es desarrollada en el trabajo de Mould [Mou07]. Para ello utiliza un grafo con pesos que se crea a partir de la imagen de entrada. Los pesos dependen del tono y el gradiente entre vecinos. A partir de este grafo se hacen crecer las zonas que contienen información más significativa para las características de la imagen.

La primera simulación de una variante del punteado tradicional llamada *hedcut*, en la que se resaltan los patrones a diferencia del tradicional, es llevada a cabo por Kim et al. [KSL*08]. Para ello, partiendo de la imagen de entrada generan un campo de distancias, esto es píxeles que se encuentran a la misma distancia dados unos píxeles iniciales con cierta característica, por ejemplo el tono. Esto permite crear una líneas que siguen la característica deseada. Dichas líneas se usan para generar los puntos cuyo tamaño depende también del tono. [Wan10] es una variante en la que además se usan las lineas isófotas.

En los primeros trabajos, el interés se centró en el posicionamiento. Tanto es así que se tiene por normal el simular los puntos por círculos de color negro y en algunos casos ni se cambia el tamaño. Es el trabajo de Kim et al. [KMI*09] el primero en considerar la información aportada por los artistas como muy relevante. Así, a partir de una obra original extrae pedazos que tienen tonos diferentes y a partir de los mismos se generan nuevas versiones que son usadas para obtener la imagen final. De esta manera se mantiene el aspecto y el estilo del dibujante.

Mientras que este último trabajo aporta imágenes muy realistas, tiene problemas de combinación de los pedazos y además está limitado por el tamaño de las imágenes de entrada. Martín et al. [MALI10, MALI11] plantea, a partir del estudio de obra manual, que los puntos no deben tratarse como círculos negros sino que en realidad varían en forma, tamaño y tono. Para conseguir un mayor realismo se plantea que la reproducción debe ser lo más fiel al original en todos sus aspectos. Eso implica el uso de puntos escaneados a una cierta resolución para ser usados para producir un cierto tamaño de salida. Para el método de posicionamiento utiliza un algoritmo de medios tonos basado en difusión de error ([Ost01]). El uso de tonos de gris permite simular mejor la física de la difusión de la tinta.

En el trabajo de Arroyo et al. [AML10a] se presenta un método que trabaja con una función de densidad para controlar que puntos se deben dibujar, aplicando un método de Monte Carlo para la generación de los mismos. En el programa, de forma interactiva, el usuario va marcando las características que desea para el dibujo final.

Li and Mould [LM11] usan el método desarrollado en [LM10] como base para el método de punteado. La idea consiste en producir una imagen de punteado mediante la modificación de una imagen de medios tonos, eliminación de píxeles. Además permite generar otros estilos como el rayado.

3. Punteado tradicional

Antes de exponer los detalles del método que se ha desarrollado, es conveniente tener una idea general de en qué consiste la técnica de punteado tradicional que vamos a simular. Como se ha indicado al principio, el punteado consiste en producir una imagen sólo usando puntos generados con una plumilla. Es importante distinguir que hay dos componentes, una artística y otra instrumental. La primera tiene que ver con todo lo relacionado con el artista y sus decisiones, esto es, por qué dibuja o no alguna zona, por qué realza una parte y difumina otra, etc. También se relaciona con el estilo y la forma de representar. La otra componente es la instrumental, cómo y con qué se crean los puntos, su tamaño y aspecto, la plumilla y el papel usados.

Actualmente, no se ha avanzado lo suficiente como para reemplazar los procesos cognitivos del artista (aunque están empezando a aparecer trabajos en esta línea basados en *deep learning*, p.e. [GEB16, GEB*17]). Por tanto, la parte artística tiene que seguir siendo una parte controlada y definida por el mismo. Por otro lado, el ordenador puede calcular las posiciones de los puntos y sus características visuales, creando distribuciones y reproduciendo puntos muy similares a los generados a mano, aunque limitada por las capacidades de los dispositivos de salida, especialmente las impresoras láser ([MdSRI15]).

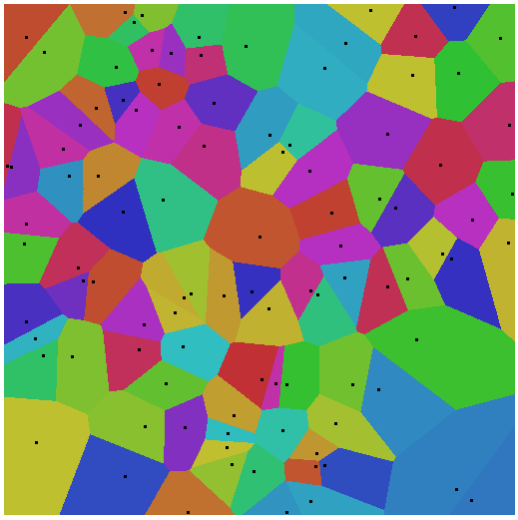
4. Punteado digital

El punteado digital consiste en la producción de imágenes con puntos usando ordenadores y dispositivos de entrada y salida. Se puede pensar que es una variante del punteado tradicional pero en realidad es una generalización pues incluye más posibilidades, con restricciones y ventajas respecto a la técnica manual.

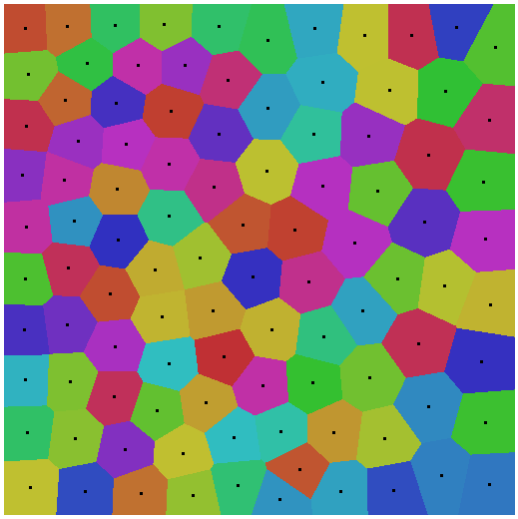
En concreto, el uso de ordenadores es especialmente potente para el cálculo de posiciones y para la representación de los puntos. Con ordenadores actuales es posible generar cientos de miles de millones de puntos por segundo. Esto implica que una actividad que necesita de muchas horas e incluso días se puede realizar incluso en tiempo real, abriendo nuevas posibilidades. Otra ventaja es la posibilidad de usar modelos 3D, lo cual permite la realización de animación con el punteado, explorando alternativas que no eran viables con el punteado tradicional.

Podemos afirmar, sin lugar a dudas, que el uso del ordenador para simular el punteado ha servido para ampliar las posibilidades de la técnica original. Tanto es así que dicha extensión ha hecho que en ocasiones se haya producido una mezcla confusa entre distintas áreas, como por ejemplo es el caso del punteado y el medio tono, asimilando que son cosas iguales, o entre punteado y generación de distribuciones de puntos, extrapolando las ventajas de un proceso a una técnica. En ambas situaciones es conveniente exponer que no siempre es así, pues el punteado es una técnica artística y el medio tono es una técnica reprográfica.

Dicho esto, pasamos a mostrar cuales han sido las técnicas y algoritmos que más se han usado para posicionar los puntos, la tarea principal en punteado, y conseguir modificar el aspecto de los puntos, tarea que actualmente ha ganado importancia cuando el objetivo es una reproducción fidedigna.



(a) Distribución inicial



(b) Resultado final

Figure 1: *Ejemplo de distribución de puntos en un diagrama de Voronoi en su estado inicial y al finalizar la distribución de energía.*

4.1. Posicionamiento

El posicionamiento de los puntos resuelve el siguiente problema: dada una imagen de entrada, seleccionar las posiciones dentro de la misma tales que al ser representadas por puntos, se obtiene una imagen que es identificable y que tiene características similares a la de la obra hecha a mano. Es importante resaltar la componente artística porque si no se puede errar y mezclar con el objetivo de la

técnica del medio tono que es conseguir la producción de tonos de gris en dispositivos monocromáticos. Esto es, no tiene sentido usar el punteado para sólo simular el tono. Los puntos son visibles desde distancias normales de lectura (30-70 cm) por lo que no cumplen el requisito de los medios tonos de que se produzca una fusión en el sistema visual. Es normal coger una fotografía en niveles de gris y convertirla a medios tonos para "reproducirla" pero no tiene sentido usarla sin modificar para "representarla".

Por tanto, la distribución de los puntos es muy importante para obtener una buena imagen de punteado, no sólo debe representar a la imagen de entrada sino que además debe cumplir con otras condiciones. Es particularmente importante que no se aprecien patrones (aunque esta es la característica principal de la variante llamada *hedcut*). Otras condiciones a tener en cuenta por el posicionamiento son la necesidad de solapar los puntos o la de modular el tamaño de los puntos dependiendo del tono de entrada.

Existen numerosas técnicas para obtener el posicionamiento siendo las más importante las que se comentan a continuación:

- Basados en diagramas de Voronoi
  El método de los diagramas de Voronoi se puede considerar como el primero que se usó en punteado ([DHvOS99,DHvOS00]). Los diagramas de Voronoi establecen una partición del espacio, en nuestro caso 2D, de tal manera que dados un conjunto de sitios, puntos en el espacio, se crean las zonas o regiones propias de cada punto p, de tal manera que los puntos incluidos en dicha zona tienen a p como sitio más cercano (ver Figura 1(a)). Si entendemos el área de cada región como su energía, es posible repartir el total de la energía de todas las regiones de tal manera que se tienda a una distribución uniforme. Para ello se deben mover los sitios. El método más usado es el algoritmo de Lloyd [Llo82]. El mismo hace que se muevan los sitios a los centroides de cada región. Para ello se haya la media de las posiciones. El diagrama obtenido se llama Diagrama de Voronoi Centroidal (ver Figura 1(b)).
  Este procedimiento se puede usar para crear una imagen de punteado: se obtiene una distribución inicial de sitios a partir de la imagen de entrada y se calcula el DVC. Normalmente se ha usado una imagen de medios tonos como distribución inicial. Una variante más potente desarrollada por Secord [Sec02b] son los Diagramas de Voronoi Centroidales Pesados (*Weighted Centroidal Voronoi Diagram*). En este método se consigue que el movimiento de los centroides esté controlado por el contenido de la imagen de entrada al utilizarla como "peso": las posiciones en zonas oscuras tienen más importancia (peso) que las zonas claras. Esto hace que se acumulen más puntos en las zonas oscuras que en las claras, comportamiento que coincide con lo que se puede ver en la obra tradicional.
  Este método y sus variantes ([HHD03,Hil06,DKLS06,SLK05]) se han usado con asiduidad, pero hay que indicar que tiene algunas desventajas, siendo la principal el que puede producir patrones si no se controla la terminación del proceso de movimiento de los centroides([BSD09]). El mismo Balzer [BSD09] propone una variante para controlar este efecto.
- Funciones de probabilidad
  Otra forma de obtener las posiciones ha sido mediante funciones de probabilidad. En este caso, la función indica la probabilidad de que un cierto punto se dibuje o no. Por ejemplo, Secord [SHS02] usa una función de densidad de probabilidad (*Probability Density Function*) generada a partir de la imagen de entrada. La misma tiene el objetivo de redistribuir el conjunto de puntos que se genera aleatoriamente como distribución inicial.
  Otro ejemplo es Arroyo et al. [AML10b, AML10a], quien usa una función de probabilidad que se genera a partir de la imagen de entrada en función del tono de los píxeles: los más oscuros tienen mas probabilidad de ser dibujados que los más claros. Esta función controla un proceso de muestreo basado en Monte Carlo. El sistema permite cambiar de forma interactiva los parámetros que controlan la distribución y por tanto el resultado final.
- Medios tonos
  Los algoritmos de medios tonos ha sido usados bien para generar las posiciones de los puntos directamente ([MALI10]) o bien para calcular la posición inicial de los puntos ([DHvOS99,DHvOS00,SHS02]).
  Aunque existen distintas alternativas, relacionadas con el tamaño del punto a reproducir, desde el uso de matrices de píxeles para representar el punto, lo que se llama modulación de amplitud (*amplitude modulation halftoning; clustered-dot dithering*) [Agf94], hasta el uso del píxel para representar el punto, lo que se llama difusión de error (*error-diffusion halftoning; frequency modulation halftoning*) [LAG99]. El principio de funcionamiento se puede ver en [FS75].
- Teselas
  Las teselas son las piezas que permiten crear un recubrimiento de algún espacio sin solapes ni huecos, en nuestro caso es un espacio bidimensional. En general, se usan recubrimientos que usan piezas semejantes que se repiten formando un patrón, como por ejemplo, el uso de cuadrados o hexágonos en casos sencillos, pero también es posible hacerlo sin que se produzca repetición, como es el caso del teselado (*tiling*) de Penrose o el de Wang [Wan61,Wan90].
  El principio de uso de las teselas se basa en que pueden recubrir el espacio: si se pone, por ejemplo, un punto en el interior de cada tesela, es posible realizar un punteado regular. Si disponemos de alguna manera de controlar el que se pinte el punto en algunas teselas y en otras no, tendremos el mecanismo para poder representar imágenes que puedan ser reconocidas. Si a esto añadimos el que es posible hacer el que cierto tipo de teselas puedan dividirse recursivamente en otras más pequeñas, lo cual permite controlar la densidad, tendremos un mecanismo general para punteado.
  Un ejemplo del uso de teselas para generar distribuciones de puntos, pero no centrándose en la generación de punteado la podemos encontrar en el trabajo de Ostromoukhov [Ost07]. Un trabajo orientado al punteado con la idea de poder producir un zoom infinito es el desarrollado por Kopf et al. [KCODL06].
- Síntesis de texturas
  La idea de la síntesis de texturas se basa en tomar la imagen de entrada y producir otra u otras imágenes que son semejantes pero no iguales. Para ello se utilizan distintos procedimientos, pero en general se establece un parecido estadístico en algunos casos a nivel de píxel y en otros a nivel de elementos de más alto nivel. Un ejemplo son los trabajos de Barla et al. [BBT*06] y Hurtut et al. [HLT*09].
  Más relacionado con el punteado, y suponiendo un gran avan-

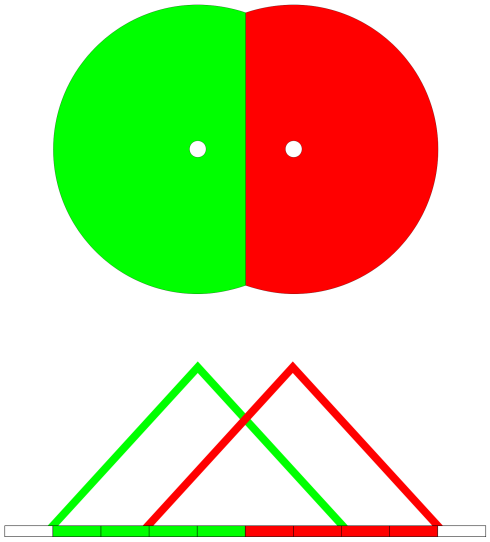


Figure 2: *Principio de funcionamiento del método de Hoff [HKL*99]. El dibujado de los conos con la eliminación de partes ocultas realizada por el z-buffer permite obtener las distintas regiones cuando se observan desde una posición cenital*

ce en la reproducción del punteado está el trabajo de Kim et al. [KMI*09]. En este caso, los puntos, solo de color negro, son más parecido a los reales, pero se generan en zonas o parches correspondientes a distintos tonos, lo cual hace que se encuentren problemas en las uniones entre zonas.

Con respecto a la calidad de la distribución de puntos, se considera que las mejores distribuciones son aquellas que poseen en su distribución espectral lo que Ulichney [Uli88] estableciera como ruido azul. Un ejemplo son las distribuciones por disco de Poisson [DW85, EDP*11] o lanzamiento de dardos [Coo86]. Estas distribuciones se han usado para texturado, visualización basada en puntos, métodos de Monte Carlo, etc. Uno de los usos que se ha dado ha sido para generar punteado. A pesar de usarse comúnmente, hay indicar las limitaciones que esta forma de evaluación cuantitativa tiene pues se realiza sobre zonas pequeñas y regulares, y se desarrolló para los medios tonos no para punteado tradicional. Por tanto, se puede considerar como una opción válida para punteado digital, aunque no esté comprobada su veracidad con respecto a la reproducción de punteado tradicional.

4.2. Generación de los puntos

Para la generación de los puntos hay menos variedad, y además, durante mucho tiempo ha sido un aspecto secundario, considerando que un punto era un círculo de color negro.

- Puntos de OpenGL
  La primitiva más usada ha sido el punto circular de color negro generado por OpenGL. En algunas ocasiones se ha modulado el tamaño. También en algunos casos se ha utilizado la posibilidad del *blending* para suavizar la forma. El tamaño ha estado restringido por el propio OpenGL pero esto no ha solido ser un problema.
- Síntesis de texturas
  La generación de puntos mediante texturas es un proceso que también engloba el posicionamiento de los mismos, y ha sido tratado en la Subsección 4.1.
- Basadas en ejemplos
  El uso de puntos escaneados se basa en el principio de que no hay nada más real que la propia realidad. En el trabajo de Martín et al. [MALI10] se presentó la idea de que usar puntos circulares de color negro no se correspondía con la realidad, sino que los mismos podían tener distintas formas y tamaños, y además que estaban compuestos por una gradación de grises. Para demostrarlo se escanearon puntos de dibujos hechos a mano de punteado a una alta densidad, lo cual permitía ver fácilmente esta característica. En un trabajo posterior, además se ha mostrado que no solo se puede ver en las aplicaciones, sino que se percibe en distancias normales de lectura y con cierto tamaño de pluma y papeles ([MdSRI15]).
  El procedimiento consiste en crear un conjunto de puntos de distintos tamaños mediante el escaneado de los puntos reales a distintas densidades (300, 600 y 1200 ppi). Una vez se tienen los puntos, el dibujado consiste en reproducir la matriz que representa al punto en la posición deseada. Dado que los puntos escaneados tiene un tamaño mayor que un píxel, desde 4×4 a 8×8 píxeles a 300 ppi hasta 16×16 a 32×32 píxeles a 1200 ppi, es necesario un reajuste del tamaño de la imagen de salida. Además, otro de los principios que se presentaron fue que los artistas producen solapado de los puntos bien porque la zona es oscura y la densidad de puntos es grande, bien por la imprecisión en el dibujado. Esta característica se implementa mediante un parámetro que controla el factor de solapamiento, lo que llaman factor de empaquetado. Al tener los puntos con niveles de gris, es posible reproducir el efecto del solape y hacer que se simule una acumulación de tinta, permitiendo obtener un resultado mucho más natural.

5. Implementación

El objetivo de este artículo es generar un sistema que permita, o bien a partir de imágenes de entrada o bien mediante herramientas de dibujo, la creación de imágenes realistas de punteado en tiempo real o interactivo. Para alcanzar los dos objetivos principales, la velocidad y el realismo, es necesario un método de posicionamiento rápido y efectivo, así como una generación de puntos realista.

A pesar de que existen otros métodos que permiten la generación de una distribución de puntos con propiedades de ruido azul e incluso operando en tiempo real, nos hemos decantado por el algoritmo de los Diagramas de Voronoi Centroidales Pesados porque tienen la ventaja de poder alinear los puntos en ciertas características visuales como bordes y siluetas, capacidad que no se encuentra o es muy reducida en otros métodos. Esta ventaja implica el problema de cierta regularidad en las zonas con poca variación. Dado que el proceso es costoso y lento, planteamos una nueva versión basada en GPU que incluya los últimos avances usando OpenGL 4.5. Para el realismo se usarán puntos escaneados basado en ejemplos, así como el procedimiento de mezcla desarrollado por Martín et al. [MALI10]).

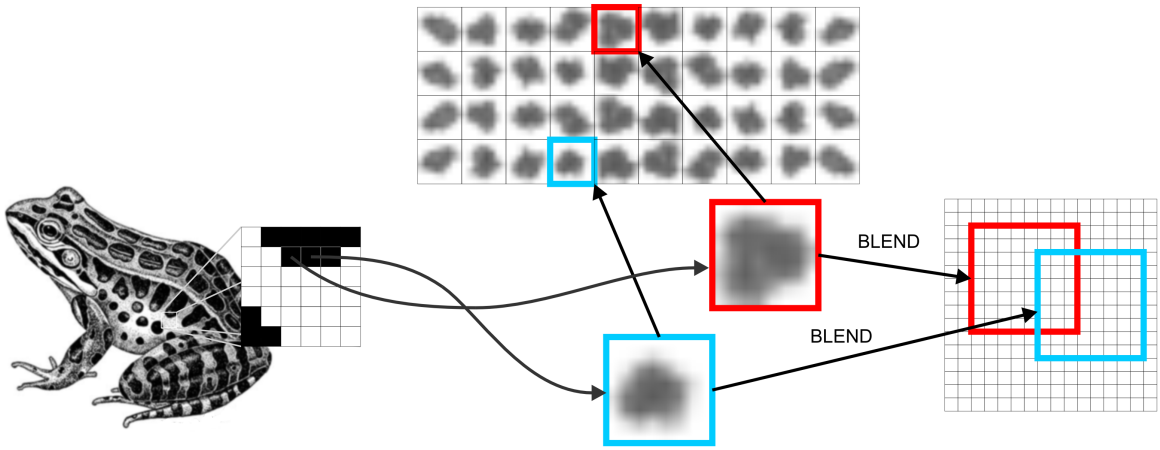


Figure 3: *Procedimiento para el dibujado de los puntos. De izquierda a derecha. A partir de la imagen de entrada se genera de forma aleatoria el puntos de la textura que que se van a dibujar. A partir de la textura se calculan las coordenadas de textura, y usando información de la imagen de entrada, se calculan las posiciones de los vértices que se corresponden con la textura. Finalmente se dibujan aplicando una función de mezcla para simular la acumulación de tinta.*

5.1. Diagramas de Voronoi Centroidales Pesados

Para la implementación del Diagrama Centroidal de Voronoi hay que distinguir entre dos procedimientos: la obtención de las regiones y el cálculo de los nuevos centroides. Para la implementación de las regiones en GPU hemos elegido la aproximación usada por Hoff et al. [HKL*99]. Existen otras implementaciones que usan la misma idea ([VSCG08]) o una aproximación diferente ([RLW*11]).

Para obtener las regiones del DVC discreto, se dibuja un cono por cada región, haciendo que el ápice coincida con la posición del sitio. Dado que se usa un proyección paralela y el z-buffer, la intersección entre conos identifica los puntos/píxeles que están a la misma distancia (ver Figura 2). Cada cono se aproximada por una pirámide de n lados de radio R. R se calcula de tal forma que una sola pirámide pueda cubrir toda la imagen: $R = \sqrt{Ancho^2 + Alto^2}$. Para el cálculo de n se hace que el error máximo en la aproximación sea de 1 píxel, $e = 1$. Se calcula el ángulo correspondiente $\alpha = 2 * \arccos(\frac{R-e}{R})$, y por tanto queda que $n = (round)(\frac{2\times\pi}{\alpha})$. Cada cono se dibuja con un color correspondiente a un identificador.

Para calcular el nuevo centroide se miran todos los píxeles que pertenecen a una región acumulando sus posiciones para finalmente hallar la media dividiendo por el número de píxeles de dicha región. La identificación de cada píxel se realiza con la conversión del color al identificador. Por tanto, para cada píxel se obtiene su identificador ID, y su posición en la imagen, x y y. En un vector con una posición para cada cono dibujado se guardan los valores:

$$Region[ID].x = Region[ID].x + x$$
$$Region[ID].y = Region[ID].y + y$$
$$Region[ID].z = Region[ID].z + z$$

Al final del proceso, se obtienen los centroides dividiendo por el número píxeles de cada región:

$$Region[ID].x = \frac{Region[ID].x}{Region[ID].z} \quad Region[ID].y = \frac{Region[ID].y}{Region[ID].z}$$

Para calcular la versión en la que se pesan los centroides, es necesario que para cada píxel de una región se obtenga el píxel correspondiente de la imagen de entrada. Dicho tono es usado para modular la contribución de la posición haciendo que tonos oscuros contribuyan más que tonos claros. Si codificamos el tono como un entero entre 0 y 255, siendo 0 negro y 255 blanco, el peso se calcula así: $peso = (255 - tono)/255$. En este caso la media se obtiene dividiendo por la suma de los pesos de todos los píxeles de la misma región.

Para nuestra implementación de las regiones hemos usado instancias (OpenGL 3.1, marzo 2009). El factor más importante para acelerar el proceso es una correcta elección del radio de cono: cuanto más pequeño más rápido, pero con la condición de no dejar ninguna zona sin cubrir. Hemos aplicado una heurística sencilla basada en dividir el radio en base al logaritmo en base 2 del número de puntos. Para la implementación de los centroides hemos hecho uso de los *Shader Storage Buffer Object* y de las operaciones atómicas (OpenGL 4.3, agosto 2012) que permiten resolver los problemas de coherencia en memoria. Para el proceso completo se hace uso de la técnica de intercambiar los buffers en cada pasada.

5.2. Puntos escaneados

Para el dibujado de puntos hemos elegido el uso de puntos reales escaneados, aunque se ha añadido la opción de dibujar círculos para comparar. Para poder dibujarlos mediante la GPU hemos recurrido a convertirlos en una textura, haciendo que cada punto se dibuje como un cuadrado descrito mediante dos triángulos, asignándole las coordenadas de textura de forma aleatoria dependiendo del tamaño de punto que se desea dibujar y de la variedad de puntos para dicho tamaño (ver Figura 3). La textura sólo se muestrea una vez por fragmento. Se usa una interpolación GL_NEAREST. Para el tamaño de punto se puede elegir una selección basada en un valor aleatorio o

N° puntos	512x512			1024x1024		
	CPU-GPU	GPU sin h.	GPU con h.	CPU-GPU	GPU sin h.	GPU con h.
128	0.1906 / 5.24	0.0215 / 46.38	0.0234 / 42.66	0.8864 / 1.12	0.1034 / 9.67	0.0644 / 15.50
500	0.1790 / 5.58	0.0352 / 28.38	0.0222 / 44.89	1.2622 / 0.79	0.1150 / 8.69	0.0589 / 16.95
1000	0.2087 / 4.79	0.0400 / 24.98	0.0168 / 59.41	1.5849 / 0.63	0.2353 / 4.24	0.0586 / 17.04
3000	0.1873 / 5.33	0.0674 / 14.82	0.0162 / 61.42	1.8761 / 0.53	0.3270 / 3.05	0.0543 / 18.38
10000	0.2029 / 4.92	0.0915 / 10.92	0.0172 / 58.03	2.8462 / 0.35	0.5302 / 1.88	0.0531 / 18.82
32768	0.2130 / 4.69	0.2337 / 4.27	0.0387 / 25.81	5.2246 / 0.19	1.2421 / 0.80	0.1535 / 6.51

Table 1: *Resultados en milisegundos y fotogramas por segundo para distintas cantidades de puntos, dos tamaños de imagen, para la versión mixta CPU-GPU con uso de heurística, para la versión con GPU sin heurística, y la versión GPU con heurística.*

se puede modular en función del tono de la imagen de entrada, haciendo que a tonos más oscuros le correspondan tamaños mayores y viceversa. Para cada tamaño de punto existen distintas variedades para evitar los patrones visuales.

Este procedimiento es valido si no hay solape entre puntos o si los mismos son todos de color negro, ya que sólo se tiene en cuenta la escritura en el *frame buffer*. Si se desea utilizar puntos con niveles de gris y simular la acumulación de tinta, esto implica que tiene que haber una operación de lectura-combinación-escritura sobre el *frame buffer*. Esto es justamente lo que se hace con las operaciones de *blending*. En general, esta funcionalidad se usa para simular las transparencias, pero nosotros la vamos a usar para simular la acumulación de tinta. Para ello la operación de combinación se implementa mediante una multiplicación: $Tono_{final} = Tono_1 \times Tono_2$. En nuestro caso, hay que tener en cuenta que la operación se realiza sobre el *frame buffer*, con lo cual la fórmula queda así: $Tono_{destino} = Tono_{destino} \times Tono_{fuente}$.

```
glBlendEquation(GL_FUNC_ADD);
glBlendFunc(GL_DST_COLOR,GL_ZERO);
glEnable (GL_BLEND);
```

La explicación es la siguiente. Dado que queremos multiplicar la fuente con el destino, debemos usar la función GL_FUNC_ADD. En su forma general, la combinación se define de la siguiente manera $Tono_{destino} = Tono_{fuente} * Factor_{fuente} + Tono_{destino} * Factor_{destino}$. Si se hace que $Factor_{fuente}$ sea igual que $Tono_{destino}$ y el $Factor_{destino}$ sea igual a cero, el resultado que se obtiene es el deseado: $Tono_{destino} = Tono_{fuente} * Tono_{destino} + Tono_{destino} * 0$

6. Resultados

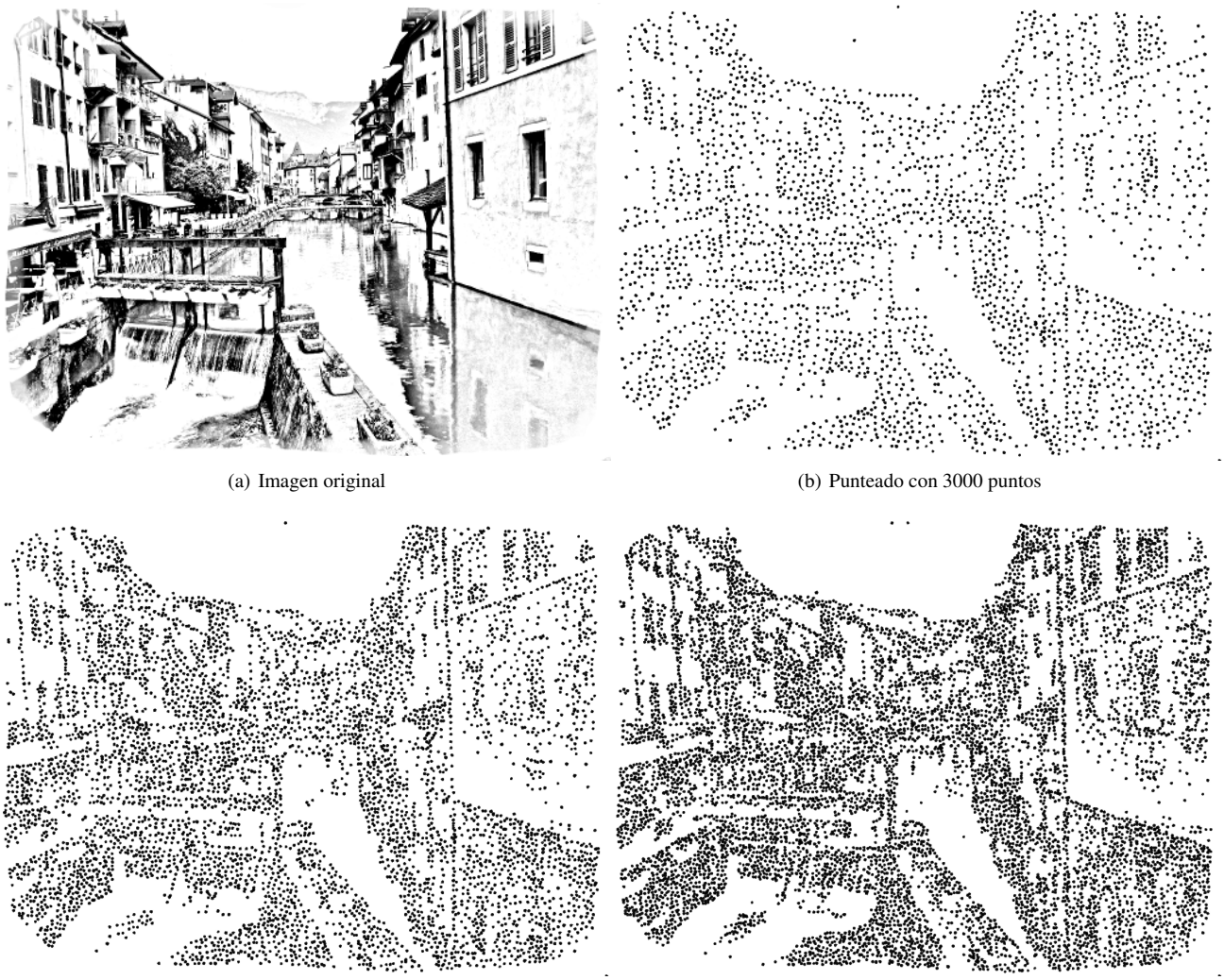
Para la obtención de tiempos se han usado dos versiones: una mixta basada en CPU-GPU y otra la que se ha expuesto sólo con GPU. En el caso de la CPU-GPU, no se usan instancias sino que los conos se convierten en una lista de triángulos, y es la CPU la que calcula los centroides, así como el dibujado de los puntos. Se ha utilizado un equipo con un procesador i5 6600k a 4GHz, una tarjeta gráfica GTX 960 y con 16GB de RAM a 2400MHz. Los datos para distintas configuraciones se pueden ver en la Tabla 1. Se ha usado una distribución uniforme de puntos.

De dichas cifras cabe destacar la diferencia que se produce al pasar de la versión CPU-GPU con heurística a la versión GPU con heurística en la que se pueden encontrar mejoras de hasta un factor de 12 en imágenes de 512x512 y de hasta un factor 53 en imágenes de 1024x1024. Con respecto al uso o no de la heurística en la versión GPU, se puede observar una considerable mejora, por lo que se convierte en un factor a optimizar aún más.

En comparación con método de Vasconcelos et al. [VSCG08], nuestra versión difiere en el uso de instancias (en 2008 no existían) y en el cálculo de los centroides (tampoco existía la funcionalidad que hemos usado (SSBO y operaciones atómicas) por lo que tienen que usar lo que llama *multi-dimensional reduction*). Los resultados extremos para imágenes de 512x512 se producen con 128 regiones (71.8 ms / 13.9 fps) y 98304 regiones (18.0 ms / 55.4 fps); para imágenes de 1024x1024 píxeles los extremos son 128 regiones (305.7 ms / 3.2 fps) y 32768 regiones (104.4 ms / 9.5 fps). En relación a los tiempos que obtienen, es difícil hacer una comparación pues no disponemos del programa y la información suministrada es muy limitada. Por ejemplo, no se indica el criterio de parada ni el número de pasadas que se utilizan; tampoco la distribución de puntos. En cualquier caso parece que la versión que presentamos es mejor, no sólo gracias a las mejoras en el hardware sino a también al software.

En comparación con el método de Rong et al. [RLW*11], en el mismo se indica que el tiempo que necesita para 1000 regiones en una imagen-textura de 512x512 es de 0,550 segundos para densidades constantes y 2,529 segundos para no constantes (similar al pesado). También hace referencia a que el método de Vasconcelos et al. necesitaría 1,416 segundos para las mismas condiciones y 216 iteraciones, aumentando hasta 44,521 segundos si aumenta la máscara al tamaño de la imagen. Hay que indicar que en este método, el cálculo de la regiones se basa en el algoritmo *jump flooding* [RT06]. A falta de una comparación más directa, se observa que los tiempos que obtenemos son muy positivos.

Para comprobar la efectividad del método hemos generado las siguiente imágenes de prueba. En la Figura 4 podemos ver el efecto de aumentar el número de puntos. Es importante comprender que la imagen de entrada debe ser tratada para que funcionen correctamente todos los algoritmos de punteado (ver [MALI10]). En la Figura 5(b) se muestra un ejemplo en el que se puede ver el resultado de generar la imagen de punteado con puntos reales pero sin aplicar el pesado de la imagen original 5(a). En la Figura 5(c) se muestra el efecto del pesado sobre el tamaño de los puntos. En la Figura 5(d) se muestra el efecto de convertir la imagen a blancos y negros mediante un umbral. Para comprobar el cambio en el tipo de punto pasando de uno real a otro circular se muestra la Figura 5(e)



(a) Imagen original (b) Punteado con 3000 puntos

(c) Punteado con 6000 puntos (d) Punteado con 9000 puntos

Figure 4: *Ejemplo de distintos resultados obtenidos a partir de la imagen inicial consistente en una foto previamente modificada.*

en niveles de gris y aplicando pesos, y en la Figura 5(f) la misma que la anterior pero en blanco y negro.

7. Conclusiones

En el presente trabajo se ha hecho una revisión de los distintos algoritmos que hay para obtener imágenes de punteado que simulan los trabajos hechos a manos, centrándonos en los aspectos relacionados con el posicionamiento y el realismo del resultado en cuanto a la representación de los puntos. Hemos mostrado el método de los Diagramas de Voronoi Centroidales y sus ventajas e inconvenientes, siendo la primera el que permite una distribución muy realista en características de la imagen como pueden ser las siluetas y bordes, pero creando también regularidad en zonas con bajo gradiente, aunque esto último se puede controlar con el criterio de parada. Aunque el procedimiento de computo es sencillo su ejecución ha sido lenta. La aproximación de usar hardware gráfico ha permitido una gran mejora. Aunque ya existían algoritmos de GPU para realizar el cálculo del DVC, en nuestra implementación hacemos la extensión a los Diagramas de Voronoi Centroidales Pesados y además incluimos las nuevas capacidades del hardware actual permitiendo un alto rendimiento en imágenes con un gran número de puntos. Para producir imágenes realistas hemos hecho uso de puntos escaneados que mediante el uso de texturas y la mezcla realizados también en la GPU, creando una herramienta de creación de imágenes y dibujado que puede ser usado por cualquier persona para obtener imágenes de punteado de una gran calidad.

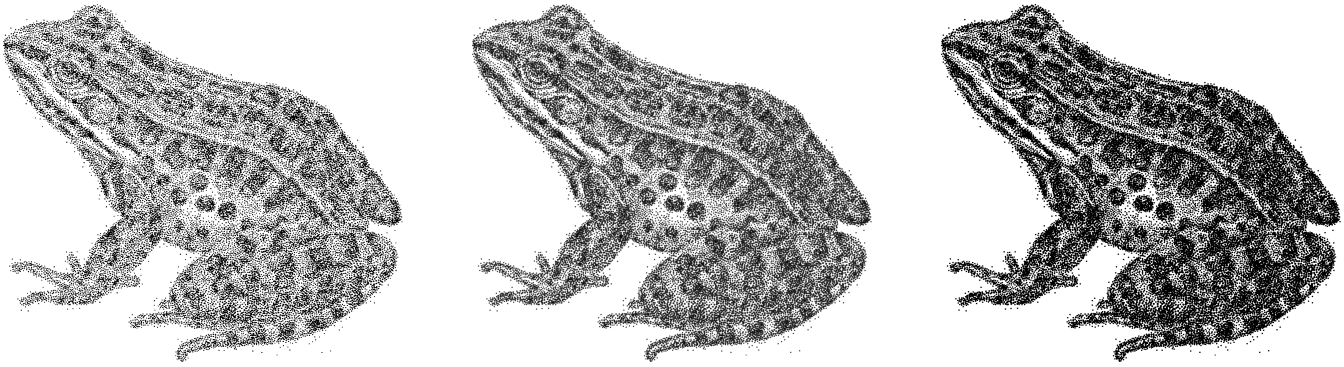
Agradecimientos

Queremos agradecer los consejos y comentarios de los revisores para mejorar el presente trabajo. Este trabajo ha sido parcialmente financiado por el Ministerio de Economía y Competitividad a través del proyecto TIN 2014-60956-R con fondos FEDER.

(a) Imagen original

(b) Punteado con puntos reales en niveles de gris sin modulación

(c) Punteado con puntos reales en niveles de gris con modulación



(d) Punteado con puntos reales en blanco y negro con modulación

(e) Punteado con círculos en niveles de gris con modulación

(f) Punteado con círculos en blanco y negro con modulación

Figure 5: *Ejemplo de distintos resultados obtenidos a partir de la imagen inicial de tamaño 1024x1024 píxeles y usando 20000 puntos (las imágenes están escaladas).*

References

[Agf94] *An Introduction to Digital Scanning*. Agfa-Gevaert, 1994. 4

[AML10a] ARROYO G., MARTÍN D., LUZÓN M. V.: A stochastic approach to simulate artists behaviour for atomatic felt-tipped stippling. In *Proc. Congress on Evolutionary Computation* (Los Alamitos, 2010), IEEE, pp. 1–8. 2, 4

[AML10b] ARROYO G., MARTÍN D., LUZÓN M. V.: Stochastic generation of dots for computer aided stippling. *Computer-Aided Design and Applications 7*, 4 (2010), 447–463. 4

[BBT*06] BARLA P., BRESLAV S., THOLLOT J., SILLION F. X., MARKOSIAN L.: Stroke pattern analysis and synthesis. *Computer Graphics Forum 25*, 3 (Sept. 2006), 663–671. 4

[BSD09] BALZER M., SCHLÖMER T., DEUSSEN O.: Capacity-constrained point distributions: A variant of Lloyd's method. *ACM Transactions on Graphics 28*, 3 (Aug. 2009), 86:1–86:8. 4

[Coo86] COOK R. L.: Stochastic sampling in computer graphics. *ACM Transactions on Graphics 5*, 1 (Jan. 1986), 51–72. 5

[DHvOS99] DEUSSEN O., HILLER S., VAN OVERVELD C. W. A. M., STROTHOTTE T.: Computer-generated stipple drawings. In *Proc. VMV* (Sankt Augustin, Germany, 1999), infix, pp. 329–338. 1, 2, 4

[DHvOS00] DEUSSEN O., HILLER S., VAN OVERVELD C., STROTHOTTE T.: Floating points: A method for computing stipple drawings. *Computer Graphics Forum 19*, 3 (Sept. 2000), 41–50. 2, 4

[DKLS06] DALAL K., KLEIN A. W., LIU Y., SMITH K.: A spectral approach to NPR packing. In *Proc. NPAR* (New York, 2006), ACM, pp. 71–78. 2, 4

[DW85] DIPPÉ M. A. Z., WOLD E. H.: Antialiasing through stochastic sampling. *ACM SIGGRAPH Computer Graphics 19*, 3 (July 1985), 69–78. 5

[EDP*11] EBEIDA M. S., DAVIDSON A. A., PATNEY A., KNUPP P. M., MITCHELL S. A., OWENS J. D.: Efficient maximal Poisson-disk sampling. *ACM Transactions on Graphics 30*, 4 (July 2011), 49:1–49:12. 5

[FJW*05] FOSTER K., JEPP P., WYVILL B., COSTA SOUSA M., GALBRAITH C., JORGE J. A.: Pen-and-ink for BlobTree implicit models. *Computer Graphics Forum 24*, 3 (Sept. 2005), 267–276. 2

[FS75] FLOYD R., STEINBERG L.: An adaptive algorithm for spatial gray scale. *SID Symoisium* (1975), 36–37. 4

[GEB16] GATYS L. A., ECKER A. S., BETHGE M.: Image style transfer using convolutional neural networks. In *Proc. CVPR* (Los Alamitos, 2016), IEEE Computer Society, pp. 2414–2423. 3

[GEB*17] GATYS L. A., ECKER A. S., BETHGE M., HERTZMANN A., SHECHTMAN E.: Controlling Perceptual Factors in Neural Style Transfer. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (July 2017). To appear. 3

[Hau01] HAUSNER A.: Simulating decorative mosaics. In *Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques* (New York, NY, USA, 2001), SIGGRAPH '01, ACM, pp. 573–580. 2

[HHD03] HILLER S., HELLWIG H., DEUSSEN O.: Beyond stippling – Methods for distributing objects on the plane. *Computer Graphics Forum 22*, 3 (Sept. 2003), 515–522. 2, 4

[Hil06] HILLER S.: *Interaktive und automatische Verfahren zur Objektanordnung in der Computergrafik*. PhD thesis, University of Konstanz, Germany, July 2006. In German. 4

[HKL*99] HOFF III K. E., KEYSER J., LIN M., MANOCHA D., CULVER T.: Fast computation of generalized voronoi diagrams using graphics hardware. In *Proceedings of the 26th Annual Conference on Computer Graphics and Interactive Techniques* (New York, NY, USA, 1999), SIGGRAPH '99, ACM Press/Addison-Wesley Publishing Co., pp. 277–286. 5, 6

[HLT*09] HURTUT T., LANDES P.-E., THOLLOT J., GOUSSEAU Y., DROUILHET R., COEURJOLLY J.-F.: Appearance-guided synthesis of element arrangements by example. In *Proc. NPAR* (New York, 2009), ACM, pp. 51–60. 4

[INC*06] ISENBERG T., NEUMANN P., CARPENDALE S., COSTA SOUSA M., JORGE J. A.: Non-photorealistic rendering in context: An observational study. In *Proc. NPAR* (New York, 2006), ACM, pp. 115–126. 2

[KCODL06] KOPF J., COHEN-OR D., DEUSSEN O., LISCHINSKI D.: Recursive Wang tiles for real-time blue noise. *ACM Transactions on Graphics 25*, 3 (July 2006), 509–518. 2, 4

[KMI*09] KIM S., MACIEJEWSKI R., ISENBERG T., ANDREWS W. M., CHEN W., SOUSA M. C., EBERT D. S.: Stippling by example. In *Proc. NPAR* (New York, 2009), ACM, pp. 41–50. 1, 2, 5

[KSL*08] KIM D., SON M., LEE Y., KANG H., LEE S.: Feature-guided image stippling. *Computer Graphics Forum 27*, 4 (June 2008), 1209–1216. 2

[LAG99] LAU D. L., ARCE G. R., GALLAGHER N. C.: Digital halftoning by means of green-noise masks. *Journal of the Optical Society of America 16*, 7 (1999), 1575–1586. 4

[Llo82] LLOYD S. P.: Least squares quantization in PCM. *IEEE Transactions on Information Theory 28*, 2 (Mar. 1982), 129–137. 2, 4

[LM10] LI H., MOULD D.: Contrast-aware halftoning. *Computer Graphics Forum 29*, 2 (2010), 273–280. 3

[LM11] LI H., MOULD D.: Structure-preserving stippling by priority-based error diffusion. In *Proc. Graphics Interface* (Waterloo, ON, Canada, 2011), CHCCS, pp. 127–134. 3

[LME*02] LU A., MORRIS C. J., EBERT D. S., RHEINGANS P., HANSEN C.: Non-photorealistic volume rendering using stippling techniques. In *Proc. VIS* (Los Alamitos, 2002), IEEE Computer Society, pp. 211–218. 2

[MALI10] MARTÍN D., ARROYO G., LUZÓN M. V., ISENBERG T.: Example-based stippling using a scale-dependent grayscale process. In *Proc. NPAR* (New York, 2010), ACM, pp. 51–61. 1, 2, 4, 5, 7

[MALI11] MARTÍN D., ARROYO G., LUZÓN M. V., ISENBERG T.: Scale-dependent and example-based stippling. *Computers & Graphics 35*, 1 (2011), 160–174. 2

[MdSRI15] MARTÍN D., DEL SOL V., ROMO C., ISENBERG T.: Drawing characteristics for reproducing traditional hand-made stippling. In *Proc. NPAR* (Goslar, Germany, 2015), Eurographics Assoc., pp. 103–115. 3, 5

[MF92] MCCOOL M., FIUME E.: Hierarchical Poisson disk sampling distributions. In *Proc. Graphics Interface* (San Francisco, 1992), Morgan Kaufmann Publishers Inc., pp. 94–105. 2

[MIA*08] MACIEJEWSKI R., ISENBERG T., ANDREWS W. M., EBERT D. S., COSTA SOUSA M., CHEN W.: Measuring stipple aesthetics in hand-drawn and computer-generated images. *IEEE Computer Graphics and Applications 28*, 2 (Mar./Apr. 2008), 62–74. 2

[Mou07] MOULD D.: Stipple placement using distance in a weighted graph. In *Proc. CAe* (Goslar, Germany, 2007), Eurographics Assoc., pp. 45–52. 2

[MPS04] MERUVIA PASTOR O. E., STROTHOTTE T.: Graph-based point relaxation for 3D stippling. In *Proc. Mexican International Conference on Computer Science* (Los Alamitos, 2004), IEEE Computer Society, pp. 145–152. 2

[MS02] MERUVIA PASTOR O. E., STROTHOTTE T.: Frame-coherent stippling. In *Eurographics Short Presentations* (Goslar, Germany, 2002), Eurographics Assoc., pp. 145–152. 2

[Ost01] OSTROMOUKHOV V.: A simple and efficient error-diffusion algorithm. In *Proc. SIGGRAPH* (New York, 2001), ACM, pp. 567–572. 2

[Ost07] OSTROMOUKHOV V.: Sampling with polyominoes. *ACM Transactions on Graphics 26*, 3 (2007), 78:1–78:6. 4

[RLW*11] RONG G., LIU Y., WANG W., YIN X., GU X., GUO X.: GPU-assisted computation of centroidal Voronoi tessellation. *IEEE Transactions on Visualization and Computer Graphics 17*, 3 (2011), 345–356. 6, 7

[RT06] RONG G., TAN T.-S.: Jump flooding in gpu with applications to voronoi diagram and distance transform. In *Proceedings of the 2006 Symposium on Interactive 3D Graphics and Games* (New York, NY, USA, 2006), I3D '06, ACM, pp. 109–116. 7

[Sec02a] SECORD A.: *Random Marks on Paper. Non-Photorealistic Rendering with Small Primitives*. Master's thesis, Department of Computer Science, The University of British Columbia, Canada, Oct. 2002. 2

[Sec02b] SECORD A.: Weighted Voronoi stippling. In *Proc. NPAR* (New York, 2002), ACM, pp. 37–43. 1, 2, 4

[SGS05] SCHLECHTWEG S., GERMER T., STROTHOTTE T.: Renderbots—Multi agent systems for direct image generation. *Computer Graphics Forum 24*, 2 (June 2005), 137–148. 2

[SHS02] SECORD A., HEIDRICH W., STREIT L.: Fast primitive distribution for illustration. In *Proc. EGWR* (Goslar, Germany, 2002), Eurographics Assoc., pp. 215–226. 4

[SIJ*07] SCHMIDT R., ISENBERG T., JEPP P., SINGH K., WYVILL B.: Sketching, scaffolding, and inking: A visual history for interactive 3D modeling. In *Proc. NPAR* (New York, 2007), ACM, pp. 23–32. 2

[SLK05] SMITH K., LIU Y., KLEIN A.: Animosaics. In *Proc. SCA* (New York, 2005), ACM, pp. 201–208. 2, 4

[Uli88] ULICHNEY R. A.: Dithering with blue noise. In *Proceedings of the IEEE* (1988), IEEE, pp. 56–79. 5

[VSCG08] VASCONCELOS C. N., SÁ A., CARVALHO P. C., GATTASS M.: *Lloyd's Algorithm on GPU*. Springer Berlin Heidelberg, Berlin, Heidelberg, 2008, pp. 953–964. 6, 7

[Wan61] WANG H.: Proving theorems by pattern recognition ii. *Bell Systems Technical Journal*, 40 (1961), 1–42. 4

[Wan90] WANG H.: *Games, Logic and Computers*. Springer Netherlands, Dordrecht, 1990, pp. 195–217. 4

[Wan10] WANG H.-Q.: Fast feature-guided stippling method based on threshold matrix. *Journal of Computer Applications 30*, 8 (2010), 2105–2107. In Chinese. 2

[XC04] XU H., CHEN B.: Stylized rendering of 3D scanned real world environments. In *Proc. NPAR* (New York, 2004), ACM, pp. 25–34. 2