Universidad Nacional del Sur

Tesis de Doctora en Ciencias de la Computación

Predicción del desempeño
de las técnicas de visualización
a partir de métricas sobre los datos

Dana Karina Urribarri

Bahía Blanca — Argentina

2014

Universidad Nacional del Sur

Tesis de Doctora en Ciencias de la Computación

Predicción del desempeño
de las técnicas de visualización
a partir de métricas sobre los datos

Dana Karina Urribarri

Bahía Blanca — Argentina

2014

Prefacio

Esta Tesis es presentada como parte de los requisitos para optar al grado académico de Doctor en Ciencias de la Computación, de la Universidad Nacional del Sur, y no ha sido presentada previamente para la obtención de otro título en esta Universidad u otras. La misma contiene los resultados obtenidos en investigaciones llevadas a cabo en el Departamento de Ciencias e Ingeniería de la Computación, durante el período comprendido entre el 12 de septiembre de 2006 y el 1 de julio de 2014, bajo la dirección de la Dra. Silvia M. Castro, Profesora Titular del Departamento de Ciencias e Ingeniería de la Computación.

Dana Karina Urribarri

UNIVERSIDAD NACIONAL DEL SUR
Secretaría General de Posgrado y Educación Continua

La presente tesis ha sido aprobada el .../.../..., mereciendo

la calificación de .....(.....)

Agradecimientos

El camino fue largo y complicado, pero finalmente llegué a destino. Gracias a todos aquellos que, de alguna forma, me acompañaron en este camino. Hicieron esto posible.

Resumen

El objetivo de una visualización es obtener una representación del conjunto de datos que ayude al usuario en la correcta interpretación de los mismos y así lograr un acertado análisis de éstos. Dado el constante crecimiento de los conjuntos de datos en diferentes y variados campos de la información, la tarea de elegir la técnica más adecuada para visualizar convenientemente los datos no es sencilla. Además, el resultado del proceso de visualización depende de todas las decisiones que se hayan tomando a lo largo de dicho proceso: un usuario inexperto es propenso a tomar decisiones equivocadas afectando negativamente la visualización obtenida y, a la larga, frustrando su experiencia con la visualización.

Si bien a la hora de visualizar conjuntos de datos pequeños no hay grandes desafíos, la situación cambia al intentar visualizar grandes conjuntos de datos: una mala decisión en cualquier punto del proceso de visualización y el resultado obtenido puede no ser satisfactorio. Una alternativa para solucionar este problema es guiar al usuario en la toma de decisiones a lo largo del proceso. Sin embargo, esta tarea no es sencilla: implica la existencia de herramientas que permitan predecir qué decisión es "más conveniente" tomar.

Una forma de elegir la decisión más conveniente es basarse en métricas sobre los datos que describan aspectos claves de la técnica y permitan *predecir* el resultado final sin necesidad de aplicar la técnica sobre los datos.

Abstract

The goal of visualization is to achieve a representation of a dataset that helps the user to interpret them correctly and achieve a proper analysis. Given the constant growing of datasets in deferent application areas, the task of choosing the more suitable technique to visualize a dataset is not easy. Besides, the result of the visualization process depends on every decision made along it: an unskilled user is prone to make incorrect decisions which affect negatively the final visualization and, eventually, frustrate the user's experience with the visualization.

Visualizing small datasets is not a big challenge, but this changes when trying to visualize big datasets: a wrong decision at any point in the visualization process and the result might not be satisfactory. A solution to this problem is to guide the user while making decisions along the process. Nevertheless, this task is not easy: it implies the existence of tools which allow the prediction of what decision if "more advisable" to make.

A way to choose the more advisable decision is using metrics over the data which describe key aspects of the techniques and allow the prediction of the final result without applying the technique to the dataset.

Certifico que fueron incluidos los cambios y correcciones sugeridas por los jurados.

Firma del Director

Índice general

Capítulo 1

Introducción

Las grandes colecciones de datos de muchas organizaciones están creciendo exponencialmente. Ejemplos de tales colecciones de datos son generados a partir de diversos campos como la simulación, los sistemas de marketing, los sistemas de salud, los sistemas atmosféricos, las redes de sensores, la Web, las colecciones de textos, etc. El problema existente es que las técnicas y las herramientas usadas en el análisis y la visualización de conjuntos de datos pequeños y medianos son inadecuadas y, en algunos casos, simplemente no son aplicables a estos grandes conjuntos de datos, en general, multidimensionales. Hay que enfrentarse, principalmente, con obstáculos tales como las limitaciones humanas, la complejidad computacional y la falta de software adecuado.

La mayoría de las técnicas de visualización actuales trabajan bien con una cantidad reducida de datos, de determinados tipos y dimensiones, y usualmente no escalan a conjuntos de datos masivos. Por lo tanto es necesario contar con técnicas que puedan adaptarse para la visualización de grandes conjuntos de datos. La exploración de estos volúmenes de datos masivos depende fuertemente de las características del sistema visual humano debido a que este es un poderoso reconocedor de patrones y, tal vez, el mejor que haya actualmente. Sin embargo, una limitación seria es la resolución del mismo, que es pobre y limitada a conjuntos de datos chicos y medianos. De esto surge la necesidad de contar con técnicas de visualización multidimensionales y escalables para trabajar con conjuntos de datos masivos.

Los análisis de escalabilidad visual que se han llevado a cabo hasta el momento no son muchos y se enfocan en problemáticas particulares. En [Weg95], por ejemplo, se enfatizan las capacidades humanas. En lo que se refiere a las técnicas de visualización, en [Kei00] se exploran sólo las limitaciones de las técnicas de visualización basadas en píxeles. En lo que se refiere a escalabilidad en visualizaciones, hay algunas propuestas interesantes sobre visualización de redes [Eic96; Eic05] y de texto [Car+99], sobre datos multidimensionales [Eic00] y sobre escalabilidad visual con grandes *displays* [Yos07; JH13]. El trabajo de Eick y Karr [EK00] es el primero que surge con el objetivo de plantear la problemática de la escalabilidad visual, particularmente en lo que respecta al rol de la visualización como un medio de lograr el entendimiento y el análisis de los datos.

1.1. Contexto de la tesis

El Laboratorio de Investigación y Desarrollo en Visualización y Computación Gráfica (VyGLab) del DCIC, UNS, ha definido un modelo de visualización unificado (MUV, ver sección 3.4) que brinda tanto al diseñador como al usuario una guía en el proceso de visualización ([Mar+03]). A partir de este modelo, uno de los objetivos del VyGLab es el diseño y el desarrollo de un modelo de visualización distribuido, que extienda el MUV y esté basado en información semántica (ver figura 1.1). Varios miembros del VyGLab se encuentran actualmente trabajando en distintas componentes del modelo de visualización. Parte del aporte de esta tesis se corresponde con definir algunas extensiones al modelo, en particular el diseño y desarrollo de un diagramado 3D de árboles y la incorporación de medidas asociadas a las técnicas que permitan asistir al usuario en la selección de la/s técnica/s más apropiada/s para visualizar su conjunto de datos.

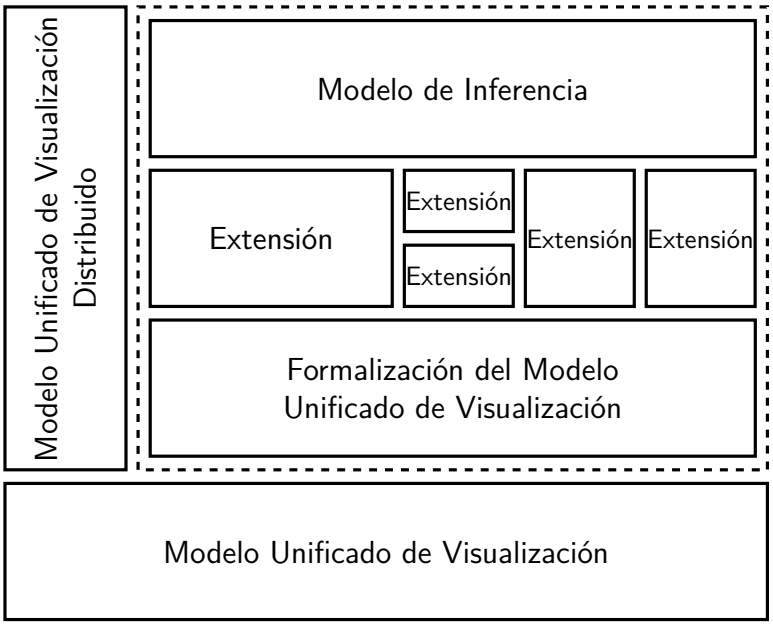


Figura 1.1: Un objetivo del VyGLab es el diseño y el desarrollo de un modelo de visualización distribuido que extienda al MUV y se base en información semántica.

1.2. Contribución de la tesis

La visualización de un determinado conjunto de datos depende de las decisiones que se toman a lo largo del proceso de visualización: el tamaño de la visualización, la técnica y sus características propias, el mapeo de colores, etc. Si bien a la hora de visualizar conjuntos de datos pequeños no hay grandes desafíos, la situación cambia al intentar visualizar grandes conjuntos de datos: una mala decisión en cualquier punto del proceso de visualización puede resultar en una visualización que no sea adecuada. Una alternativa para solucionar este problema es guiar al usuario en la toma de decisiones a lo largo del

proceso. Sin embargo, esta tarea no es sencilla: implica la existencia de herramientas que permitan predecir qué decisión es *más conveniente* tomar. Una forma de tomar la decisión más conveniente es basarse en medidas sobre los datos que describan aspectos clave de la técnica y permitan *predecir* el resultado final sin necesidad de aplicar la técnica sobre los datos.

La definición de métricas asociadas a las técnicas como forma de *predecir* qué ocurrirá con algún aspecto particular de la aplicación de dicha técnica sobre determinado conjunto de datos fue la principal motivación de este trabajo de tesis doctoral. De este modo se contará con un parámetro de decisión adicional a la hora de decidir con qué técnica/s visualizar cada conjunto de datos. Adicionalmente, esto permitirá definir métricas específicas que *ayuden* a decidir si la técnica escala visualmente para dicho conjunto de datos.

Específicamente, las contribuciones de esta tesis se agrupan en tres temas principales:

1. Diseño e implementación de un *layout* de árboles hiperbólico y multirresolución
2. Escalabilidad visual en *scatterplots* y en árboles
3. Integración de medidas de escalabilidad visual con el Modelo Unificado de Visualización (MUV)

1.2.1. Diseño e implementación de un *layout* de árboles hiperbólico y multirresolución

Uno de los problemas existentes en visualización de grandes árboles es la limitación que presenta la escalabilidad de estos sistemas: a medida que crece la cantidad de nodos se tiende a mostrar sólo la topología del árbol. De las varias técnicas de visualización de grandes árboles, una de las más apropiadas es la visualización hiperbólica de los mismos; sin embargo, las técnicas existentes basadas en este *layout* tienen limitada interactividad y no permiten la visualización de datos con alta dimensionalidad. Dado su potencial en la visualización de grandes árboles es deseable encontrar una solución para estos problemas. Como parte de esta tesis, se desarrolló el Gyrolayout [Urr+13], que extiende una de las técnicas hiperbólicas de visualización de árboles para soportar diferentes niveles de detalles, manteniendo niveles aceptables de interactividad. El desarrollo del *layout* se basó en una combinación de algoritmos de teselados discretos y en un espacio vectorial especialmente definido para modelar el espacio hiperbólico.

1.2.2. Escalabilidad visual en *scatterplots* y en árboles

Teniendo en cuenta que no es factible abarcar todo el espacio de los posibles conjuntos de datos, se decidió trabajar sobre un subespacio particularmente importante de los que se consideran grandes conjuntos de datos: nubes de puntos y árboles.

La técnica más comúnmente adoptada para visualizar nubes de puntos son los *scatterplots*. Esta técnica ha sido elegida para el desarrollo de parte de esta tesis ya que es

una de las técnicas más estudiadas y ampliamente usada en el análisis de datos. Para los *scatterplots* se definió una medida de visibilidad de los datos que permite *predecir* no solo la superposición que existirá en la visualización sino también los máximos y mínimos posibles; estos máximos y mínimos establecen los límites de cuán escalable visualmente es la técnica para dicho conjunto de datos.

Las técnicas de visualización de árboles se dividen en dos grandes grupos, las técnicas con representación implícita de arcos (*llenado de superficies*) y las técnicas con representación explícita de arcos (*nodos enlazados*); la combinación de ambas representaciones conforman las técnicas híbridas. Dentro de estas técnicas se estudiaron los *treemaps* y el Gyrolayout, como *layouts* típicos con representación de arcos implícita y explícita, respectivamente. En ambos casos se desarrollaron métricas particulares que permiten *predecir* la visibilidad de los nodos de un dado árbol y establecer los límites de la escalabilidad visual de cada *layout* con el árbol correspondiente.

1.2.3. Integración de medidas de escalabilidad visual con el Modelo Unificado de Visualización (MUV)

Tanto en el estudio como en el diseño e implementación de una visualización, resulta necesario contar con un marco de referencia como guía de diseño; esto es de gran utilidad al momento de poner en contexto las diversas técnicas. En el marco del Grupo de Investigación y Desarrollo en Visualización y Computación Gráfica (VyGLab), se cuenta con el Modelo Unificado de Visualización (MUV). El MUV es un modelo de referencia que permite reflejar tanto los estados como las transformaciones intermedias por las que deben atravesar los datos desde que ingresan al sistema de visualización hasta que son finalmente visualizados, y establece un marco conceptual independiente tanto de la técnica como del dominio de aplicación.

El desarrollo de esta tesis se encuentra vinculado a dicho modelo ya que la definición de medidas asociadas a cada técnica, describiendo los aspectos más importantes de esta, brindará una poderosa herramienta para asistir al usuario en las últimas etapas del MUV, es decir, en la elección de la técnica más apropiada para visualizar un dado conjunto de datos. En este contexto, se integraron las medidas en el proceso de visualización.

1.3. Terminología

En esta tesis se utilizan varios términos de connotación subjetiva que son utilizados habitualmente en visualización; que una visualización o una técnica sea aceptable, conveniente o efectiva dependerá del usuario y el fin que este le de a la visualización y no tanto de la visualización en sí misma. Sin embargo, resulta apropiado especificar a qué se refiere cada uno de estos términos al ser utilizados en el texto. La definición que se brinda de los siguientes términos es la definición propia en el contexto de la visualización:

- *Efectivo/a* Una visualización efectiva es aquella que le resulta de utilidad al usuario, es decir, aquella que logra satisfacer sus objetivos. Una técnica es efectiva cuando la visualización obtenida a partir de dicha técnica es efectiva, en particular, es rápida de interpretar, transmite las diferencias y conduce a menos errores.

- *Expresiva* Una visualización es expresiva cuando la decodificación visual representa toda la información existente en los atributos del conjunto de datos sin inducir información adicional.

- *Aceptable* Un valor se considera aceptable cuando se encuentra dentro de un rango especificado por el usuario. Una técnica o una visualización se considera aceptable cuando es expresiva y el usuario considera que puede resultar en una visualización efectiva.

- *Potencialmente* Una visualización o una técnica se considera potencialmente aceptable cuando algún valor derivado de estas es aceptable. En otros contextos, se utiliza como sinónimo de *probablemente* o *posiblemente*.

- *Bueno/na* Es un sinónimo de efectivo, aunque enfáticamente más débil.

- *Conveniente* Una decisión se dice conveniente cuando deriva, directa o indirectamente, en una visualización efectiva. En el texto se habla de la decisión *más conveniente*; dado un conjunto de posibles decisiones, la decisión más conveniente será alguna de las que deriven en una visualización efectiva.

- *Apropiado/a* En general se utiliza como un sinónimo de conveniente en su acepción más general. En particular, al referirse a una técnica, esta se dice apropiada para un conjunto de datos cuando, dadas las características propias de la técnica, es muy probable que resulte en una visualización efectiva.

1.4. Estructura

Esta tesis está escrita de forma que su lectura sea guiada y lo más autocontenida posible. A continuación se describe la estructura de esta y un breve detalle de lo tratado en cada capítulo.

Capítulo 1: Introducción Se introducen los objetivos y el marco en el que se desarrolló la investigación presentada en esta tesis. Además, se presentan las contribuciones de la tesis y la estructura de esta.

Capítulo 2: Escalabilidad Visual Se introduce el concepto de escalabilidad visual y la problemática asociada, teniendo en cuenta las capacidades actuales. Se analiza la escalabilidad visual de las técnicas de visualización más representativas y se describen diferentes formas generales de incrementar la escalabilidad visual de estas técnicas.

Capítulo 3: Métricas y predicción Se introduce el concepto de métrica en el campo de la visualización, junto con el marco teórico definido por diferentes autores en este tema. Además, se presentan los diferentes estados y transiciones del MUV en el que se encuadra el aporte de esta tesis. Finalmente, se plantean los beneficios de la *predicción* en el marco de este modelo y la elección de la técnica de visualización adecuada para un dado conjunto de datos.

Capítulo 4: Scatterplots Se presenta el estado del arte de la técnica de *scatterplots* que se ha realizado teniendo en cuenta la evolución de dicha técnica para representar distintos conjuntos de datos. Los *scatterplots* son gráficos 2D útiles para representar grandes conjuntos de datos; originalmente se han usado para representar conjuntos de datos escalares bidimensionales y posteriormente fueron adaptados para representar datos multidimensionales. Para esto se han presentado distintas variantes que amplían y mejoran la técnica teniendo en cuenta, en cada caso, los puntos débiles correspondientes a cada representación.

Capítulo 5: Predicción de la visibilidad en *scatterplots* Se presenta una métrica que evalúa la visibilidad de los datos teniendo en cuenta la visibilidad de los glifos en el *scatterplot*. Se definió una métrica que dados un conjunto de datos, el tamaño de la ventana y el tamaño del glifo que representará los datos, estima el porcentaje de glifos que serán siempre visibles independientemente del orden en que estos sean mostrados. Para definir y aproximar dicha métrica se experimentó con conjuntos de datos aleatorios seleccionados siguiendo distribuciones normales para ambas dimensiones de los datos.

Capítulo 6:Visualización de árboles Se introducen los conceptos de visualización implícita y explícita de árboles. Además, se presentan los *treemaps* como una técnica de visualización representativa de la visualización implícita de árboles. Finalmente, dentro de las técnicas explícitas de visualización de árboles, se propone el Gyrolayout, que es un *layout* hiperbólico 3D de árboles, generado en base a conceptos teóricos de análisis hiperbólico y a algoritmos de teselados discretos.

Capítulo 7: Predicción de la visibilidad en la visualización de árboles Se presentan dos métricas, una asociada a los *treemaps* y la otra asociada al Gyrolayout. Estas métricas permiten predecir la visibilidad de los nodos en la visualización resultante.

Capítulo 8: Conclusiones y Trabajo Futuro Se presentan las conclusiones finales y el trabajo futuro que puede desarrollarse a partir del realizado en esta tesis.

Apéndice A: Conceptos básicos de geometría hiperbólica

Apéndice B: Extensiones del Teselado de Voronoi Se presenta el Teselado esférico, baricéntrico y pesado de Voronoi aplicado tanto al Layout Esférico 3D en Larrea et al. [Lar+09] como aplicado al Gyrolayout en Urribarri et al. [Urr+13]. Para esto,

es necesario definir previamente los conceptos de Teselado de Delaunay, Teselado de Voronoi, Teselado esférico de Voronoi y Teselado esférico pesado de Voronoi.

Capítulo 2

Escalabilidad Visual

Se introduce el concepto de escalabilidad visual y la problemática asociada, teniendo en cuenta las capacidades actuales. Se analiza la escalabilidad visual de las técnicas de visualización más representativas y se describen diferentes formas generales de incrementar la escalabilidad visual de estas técnicas.

2.1. Introducción

La escalabilidad visual es definida por Eick y Karr [EK00] como la capacidad de una herramienta de visualización de mostrar efectivamente grandes conjuntos de datos, en término de la cantidad y la dimensionalidad de estos.

Idealmente, la escalabilidad es cuantificable en términos de *respuestas* y *factores*:

$$\text{respuestas} = F(\text{factores}, \text{datos})$$

donde las respuestas miden el impacto del entendimiento, los descubrimientos y las decisiones inducidas por la visualización y los factores miden las características propias de la visualización.

En este capítulo se definirá qué se consideran *grandes conjuntos de datos*. Luego, se analizarán los factores que afectan la escalabilidad visual en algunas técnicas más representativas y finalmente se presentarán algunas alternativas para incrementar la escalabilidad visual de dichas técnicas.

2.2. Grandes conjuntos de datos

Las taxonomías de datos orientadas a la escalabilidad existentes en la literatura son presentadas desde un punto de vista estadístico. Unwin et al. [Unw+06] han planteado una clasificación (ver tabla 2.1a) que divide los datos según su tamaño. Posteriormente, Wegman [Weg95] extendió la clasificación para contemplar conjuntos de datos aún más grandes (ver tabla 2.1b). Sin embargo, estas dos clasificaciones tienen en cuenta solamente el tamaño en bytes del conjunto de datos y, en general, una clasificación de los datos

basada únicamente en su tamaño, no brinda suficiente información para elegir una técnica o estrategia de visualización acorde a los datos.

Tamaño	**Descripción**	**Bytes**
Diminuto	Entra en un pizarrón	10^2
Pequeño	Entra en unas cuantas páginas	10^4
Mediano	Entra en un disquete	10^6
Grande	Entra en un disco rígido	10^8
Enorme	Necesita varios discos rígidos	10^{10}

(a) Clasificación de Huber

Tamaño	**Descripción**	**Bytes**
⋮	⋮	⋮
Monstruoso	Cintas magnéticas	10^{12}

(b) Extensión de Wegman

Tabla 2.1: Clasificación de Huber y la extensión de Wegman, basadas únicamente en en el tamaño en bytes del conjunto de datos.

Eick [Eic05] presenta el tema particular de la escalabilidad visual de redes. Además de introducir las propiedades y la estructura de las redes a partir de la teoría de grafos, presenta algunas posibles medidas de escalabilidad visual particulares para redes: número de nodos o arcos visibles y número de elementos visibles, número de componentes conexas, entre otras.

En este capítulo y a lo largo de la tesis, se considerará que un conjunto de datos no es pequeño o grande por sí mismo, sino que su tamaño dependerá del contexto. Si el espacio disponible para graficar la visualización es reducido (por ejemplo $2cm \times 2cm$), 500 datos puede ser un conjunto *grande*. Sin embargo, la misma cantidad de datos en una visualización de $10cm \times 10cm$ puede ser un conjunto *pequeño*. En el contexto de *scatterplots* Carr et al. [Car+87] consideran que un conjunto de datos es grande cuando ocurre alguno de los siguientes escenarios:

- el tiempo de "creación" de la visualización es extenso,
- el tiempo de cómputo de algunas operaciones es extenso, o
- la visualización tiene una gran superposición,

donde *tiempo extenso* se considera aquel que no es interactivo.

Extendiendo esta definición a otros contextos más generales, se puede considerar que, bajo determinadas circunstancias, un conjunto de datos es grande cuando ocurre alguno de los siguientes escenarios:

- el tiempo de "creación" de la visualización es no interactivo,
- el tiempo de cómputo de alguna operación es no interactivo, o
- la técnica de visualización ha llegado al tope de su factor limitante.

Para el análisis de la escalabilidad visual de las técnicas en función los factores que la afectan, estos se han dividido en dos grandes grupos: factores externos y factores internos. Los factores externos son los que no dependen de la técnica de visualización, mientras que los factores internos son los inherentes a la técnica de visualización. En las dos secciones siguientes se analizarán estos dos grupos de factores.

2.3. Factores externos que afectan la escalabilidad visual

Eick [Eic05] presenta diferentes factores externos que afectan la escalabilidad visual, algunos son inherentes a las personas, y otros son consecuencia del sistema computacional. Dentro de los factores más relacionados con las personas se encuentra la *percepción humana*. Si bien el humano podría percibir unos 6,5 millones de píxeles, la *resolución del monitor* varía actualmente entre 800×600 y 2560×1600 píxeles[1] alcanzando entre 480 000 y 4 096 000 píxeles. La *interactividad* es una herramienta para incrementar la escalabilidad visual, pero se ve disminuida por la incapacidad de los usuarios para navegar espacios altamente dimensionales. Las interacciones más usuales son foco+contexto, *panning* y *zooming*, selección, agregación y *brushing*. También se plantean las *estructuras de datos*, los *algoritmos* empleados y la *infraestructura computacional* (cpu, red, tasa de *rendering*) como factores relacionados al sistema computacional que afectan la escalabilidad visual.

2.4. Factores internos que afectan la escalabilidad visual

Los factores internos que afectan la escalabilidad visual son aquellos factores que limitan su expresividad y están determinados por características inherentes de la técnica de visualización. En general, los factores externos anteriormente enumerados afectan a todas las técnicas de visualización, sin embargo, no todas las técnicas se ven limitadas por los mismos factores internos. Los factores internos pueden tener su origen sólo en la naturaleza de la técnica de visualización o, estar también indirectamente asociados a un factor externo. Por ejemplo, en un gráfico de barras convencional la cantidad de dimensiones representables (1 atributo por dato) está limitada por la técnica; sin embargo, no hay límite en la cantidad de barras (datos); este límite está dado por el tamaño de la pantalla (factor externo) y el ancho de la barra (factor interno).

En esta sección se describirán algunas de las técnicas más representativas, se analizará la escalabilidad visual de cada una de ellas y los factores que la limitan. En la tabla 2.2 se resumen las limitaciones de cada una de ellas.

[1] Samsung 305TPlus 30" Wide

Técnica de Visualización	Limitación en la cantidad de ítems	Limitación en la cantidad de atributos/dimensiones	Factor Limitante
Gráficos de barras	limitado por la máxima cantidad de barras (limitado por tamaño de la ventana y ancho de la barra)	limitado por la técnica	separación entre barras, dimensionalidad acotada
Scatterplots	limitado por tamaño de la ventana, tamaño del glifo	limitado por la técnica, mejorable con características del glifo	superposición, dimensionalidad acotada
Coordenadas Paralelas	limitado por tamaño de la ventana, entrecruzamiento de polilíneas	limitado por el tamaño de la ventana	oclusión, separación entre dimensiones
Redes	limitado por la ventana, tamaño del nodo, representación del enlace, cantidad de enlaces		sobrecarga, posicionamiento de los nodos, tensión *perceptual*

Tabla 2.2: Resumen de las limitaciones de algunas técnicas de visualización.

2.4.1. Gráficos de barras

Los gráficos de barras son una colección de barras verticales (u horizontales). Cada barra puede codificar hasta dos atributos, mapeándolos a altura y color (o algún patrón distintivo); apilándolas (ver figura 2.1b) o agrupándolas (ver figura 2.1c) es posible incrementar el número de atributos representables. El máximo número de barras está limitado por el tamaño de la ventana que contenga la visualización, que a su vez se encuentra limitado por la resolución de la pantalla.

Si las barras se corresponden a más de una categoría se pueden ordenar en forma matricial y representar las barras en 3D (llamado *landscape* o *multiscape*). En este caso, la altura, el color y la forma pueden representar hasta tres atributos distintos. Dependiendo de la cantidad de píxeles utilizados para representar cada barra 3D, será la cantidad máxima de barras que podrá representarse en la imagen.

2.4.2. *Scatterplots*

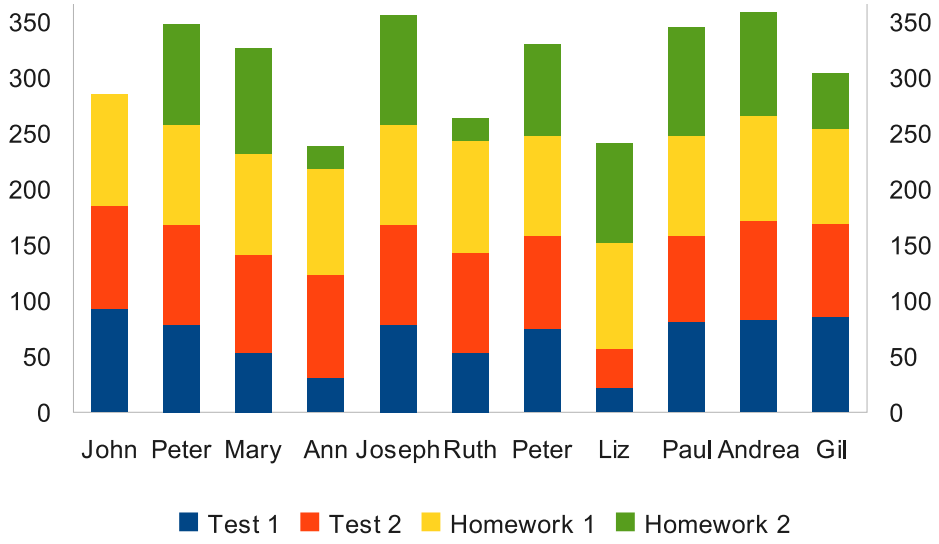
Los *scatterplots* son gráficos 2D que usan coordenadas Cartesianas para representar dos atributos de un conjunto de datos. Los datos se muestran como una colección de puntos, donde el valor de un atributo determina la posición sobre el eje vertical y el valor del otro atributo, la posición sobre el eje horizontal.

El gráfico se puede extender para representar hasta 3 atributos, si en vez de representar los datos en el plano, se representan en el espacio. También es posible mapear atributos a la representación de los puntos: forma, color (ver figura 2.2b), tamaño (ver figura 2.2c) o glifos ([Che73; Bor+13]).

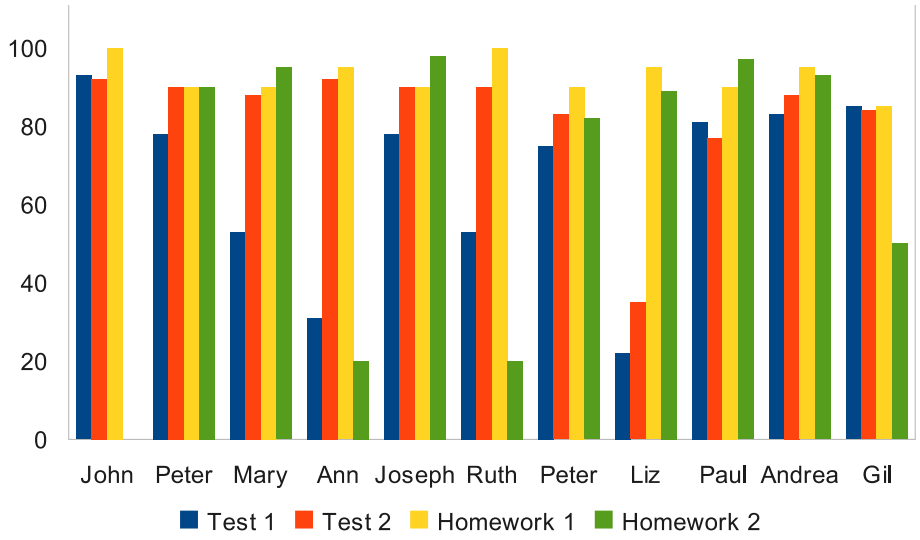
El factor limitante para la máxima cantidad de ítems representables es la superposición. Si bien un píxel es suficiente para representar un dato, logrando así un máximo de

Name	Test 1	Test 2	Homework 1	Homework 2
John	93	92	100	0
Peter	78	90	90	90
Mary	53	88	90	95
Ann	31	92	95	20
Joseph	78	90	90	98
Ruth	53	90	100	20
Peter	75	83	90	82
Liz	22	35	95	89
Paul	81	77	90	97
Andrea	83	88	95	93
Gil	85	84	85	50

(a) Tabla de notas de diferentes tests por cada alumno



(b) Notas de varios test apiladas por alumno.



(c) Notas de varios test agrupadas por alumno.

Figura 2.1: Gráfico de barras 2D con barras apiladas y en grupos. Conjunto de datos *Pupils* tomado de los ejemplos de Parvis.

tantos datos representables como píxeles haya disponibles, este escenario puede no ser satisfactorio ya que no siempre es posible individualizar cada uno de los datos visualizados. En el capítulo 5 se define una métrica de visibilidad de los glifos en un *scatterplot*.

2.4.3. Coordenadas paralelas

Las coordenadas paralelas [ID90] fueron propuestas como una técnica para la visualización de conjuntos de datos multidimensionales. Con el desarrollo de herramientas que implementaban las coordenadas paralelas, se encontró que eran prometedoras para la visualización de grandes conjuntos de datos. Esta técnica asigna un eje a cada dimensión de los datos y los ordena paralelamente en el plano. Cada dato n-dimensional es una poligonal que atraviesa los n ejes paralelos (ver figura 2.3). En la figura 2.4 se muestra la visualización de un conjunto datos (ver tabla 2.2a) con 4 atributos y 150 ítems de datos.

Aunque el número de dimensiones que se puede representar es potencialmente ilimitado, el espacio que se dispone para la representación es acotado y de resolución finita. Esto implica que un factor limitante de la escalabilidad visual de las coordenadas paralelas es la cantidad de dimensiones representables en la pantalla (ver figura 2.5a). Esta cantidad máxima se encuentra acotada por la resolución vertical u horizontal de la pantalla (dependiendo de cómo se dispongan los ejes) y del mínimo espacio necesario entre ejes para que sea posible distinguir el comportamiento de la poligonales.

Otro factor limitante es la oclusión (ver figura 2.5b). Si bien la técnica es apropiada para representar grandes conjuntos de datos multidimensionales en el plano, gran cantidad de poligonales generan una visualización confusa, donde es imposible individualizar datos o percibir patrones de comportamiento.

2.4.4. Grafos y árboles

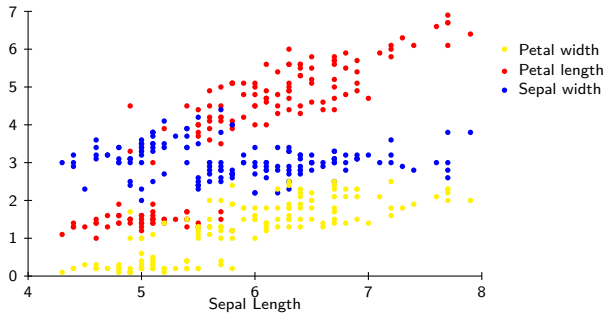
Los grafos y los árboles permiten representar tanto los ítems de datos como las relaciones existentes entre ellos.

Los nodos del grafo y sus características visuales (color, forma, tamaño, etc.) representan los datos y sus atributos, respectivamente. Los enlaces entre nodos y sus características visuales (color, forma, patrón) representan las relaciones entre los datos y sus atributos, respectivamente. Por ejemplo, si los datos son aeropuertos, compañías que realizan vuelos directos entre ellos y frecuencia de los vuelos, los nodos representarán los aeropuertos, y los enlaces los vuelos, donde el ancho del enlace puede representar la frecuencia de vuelos y el color, la compañía. En un escenario como este, los nodos estarían conectados por más de un enlace (multigrafos). El tamaño de los nodos (como un ejemplo de atributo) podría representar la importancia de la ciudad, medida en la cantidad de empresas distintas que operan allí, o la cantidad de ciudades con la que está conectada mediante vuelos directos.

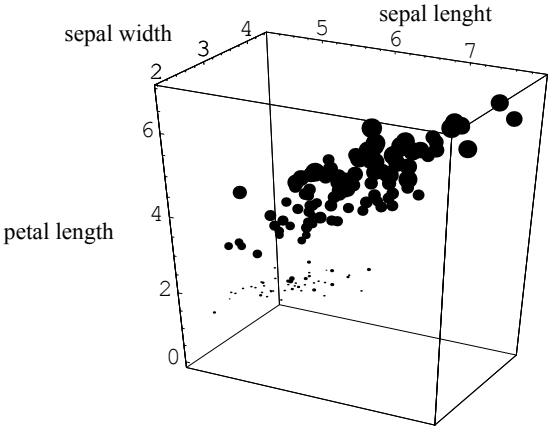
Las visualizaciones que muestran tanto nodos como enlaces son efectivas en el caso de grafos ralos. En el caso de grandes grafos aparecen tres problemas ([Eic96]) que impiden el análisis de los datos:

#	Sepal length	Sepal width	Petal length	Petal width
1	5,1	3,5	1,4	0,2
2	4,9	3,0	1,4	0,2
3	4,7	3,2	1,3	0,2
4	4,6	3,1	1,5	0,2
5	5,0	3,6	1,4	0,2
⋮	⋮	⋮	⋮	⋮
150	5,9	3,0	5,1	1,8

(a) Tabla de datos del conjunto *Iris flower*.



(b) Visualización con *scatterplot* y color.



(c) Visualización con *scatterplot* y tamaño.

Figura 2.2: Visualización del conjunto de datos *Iris flower* (150 ítems de dato) mediante *scatterplots* 2D y 3D.

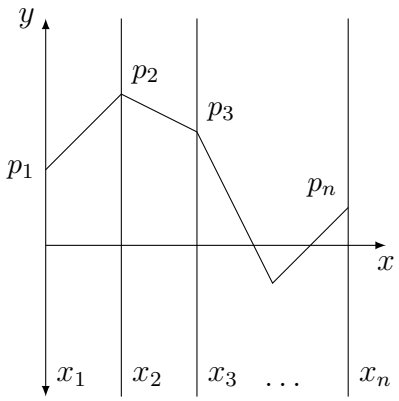


Figura 2.3: Representación de un punto en el sistema de ejes paralelos en n dimensiones.

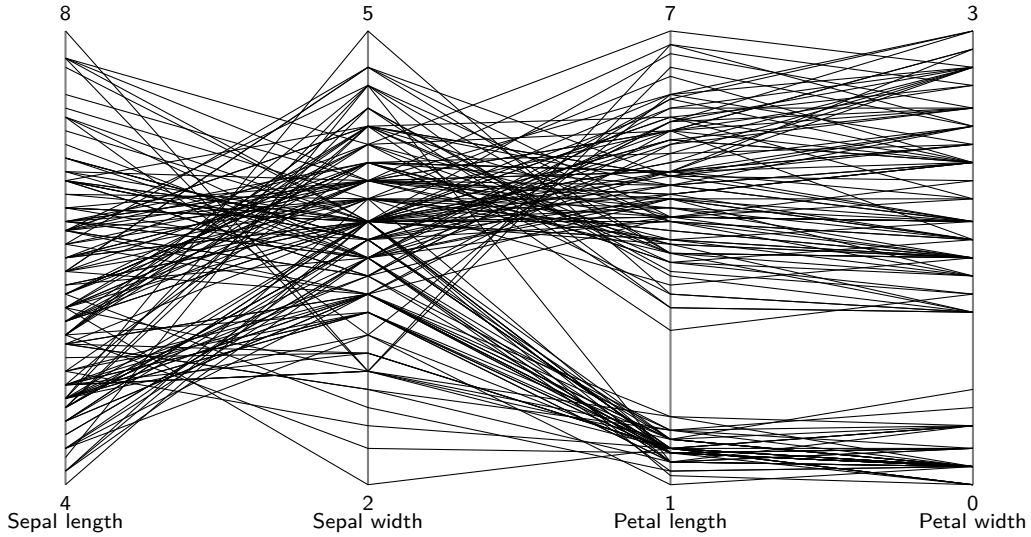


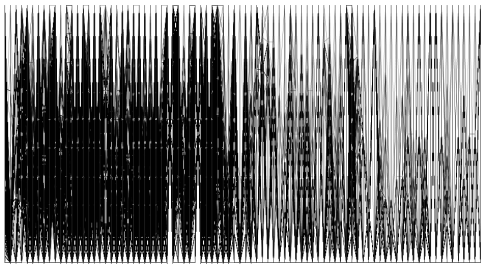
Figura 2.4: Visualización con Coordenadas Paralelas del conjunto de datos *Iris flower* de la tabla 2.2a

1. *Sobrecarga.* El gráfico se satura de información y se torna visualmente confuso.
2. *Posicionamiento de los nodos.* La interpretación de los datos a partir de la visualización es altamente dependiente de la posición de los nodos en el gráfico.
3. *Tensión perceptual*[2]*.* Los nodos cercanos entre sí se perciben relacionados aunque estén conectados con una línea corta. Por otro lado, los nodos lejanos no se perciben relacionados aunque están conectados con una larga línea dominante.

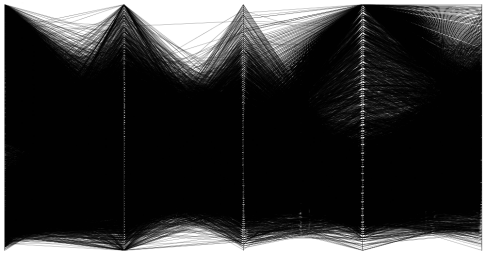
En particular, los árboles representan relaciones jerárquicas entre los datos. Todos aquellos datos que tengan una estructura jerárquica pueden ser visualizados como árboles; algunos ejemplos de estos datos pueden ser:

- Directorios generales de páginas web, como DMOZ[3] ([HC04])

[2] Palabra en inglés que significa perteneciente o relativo a la percepción.
[3] `http://www.dmoz.org/`



(a) Dependiendo de cómo se dispongan los ejes, las dimensiones de la pantalla son el factor limitante de la máxima cantidad de dimensiones representabes satisfactoriamente.



(b) Sin las interaccionesa adecuadas, la oclusión es el factor limitante de la cantidad de datos representables.

Figura 2.5: Las dimensiones de la pantalla y la oclusión limitan la cantidad de dimensiones y la cantidad de datos representables.

- Directorios particulares como el proyecto *The Tree of Life*[4] o el proyecto MACE[5]. El proyecto *The Tree of Life* reúne aquellas páginas de internet que proveen información sobre biodiversidad, las características de diferentes grupos de organismos, y sus historia evolutiva. El proyecto MACE (que es acrónimo de *Metadata for Architectural Contents in Europe*) es un proyecto que recopila información digital sobre arquitectura.
- Información genealógica como [MB05] o el proyecto personal presentado en `http://www.alexandercheek.com/genealogicalhistory.html`.

En el capítulo 6 se presentarán en mayor profundidad dos técnicas de visualización de árboles: los *treemaps* y el Gyrolayout. En el capítulo 7 se analizarán métricas sobre estas técnicas.

2.5. Estrategias para incrementar la escalabilidad visual

Eick y Karr [EK00] presentan estrategias para mejorar la escalabilidad visual. Las estrategias más relevantes que mencionan son el mejoramiento y la combinación de las metáforas visuales, el aprovechamiento de las interacciones y el permitir vistas enlazadas.

2.5.1. Mejoramiento y combinación de las metáforas visuales

Dadas las limitaciones tanto de los *displays* como del ojo humano, mejorar las metáforas visuales, o la combinación de ellas, representan una oportunidad para mejorar la

[4] `http://tolweb.org`
[5] `http://portal.mace-project.eu/`

escalabilidad visual. A continuación se brindan ejemplos de posibles mejoras a las metáforas visuales presentadas en la sección 2.4.

Gráficos de barras

En ADVIZOR [ADV10] el algoritmo de *rendering* usa una estrategia multirresolución: las barras se dibujan a la máxima resolución (incluso en 3D) cuando hay lugar suficiente en la pantalla; a medida que el usuario se aleja, las barras se dibujan con menor detalle, hasta convertirse en barras de 1 píxel de ancho. Si el usuario se sigue alejando, las barras tienden a tener menos de un píxel de ancho, y se usa entonces un indicador de *superposición*. Cuando las barras no tienen un orden intrínseco, otra estrategia incluye reordenarlas.

Scatterplots

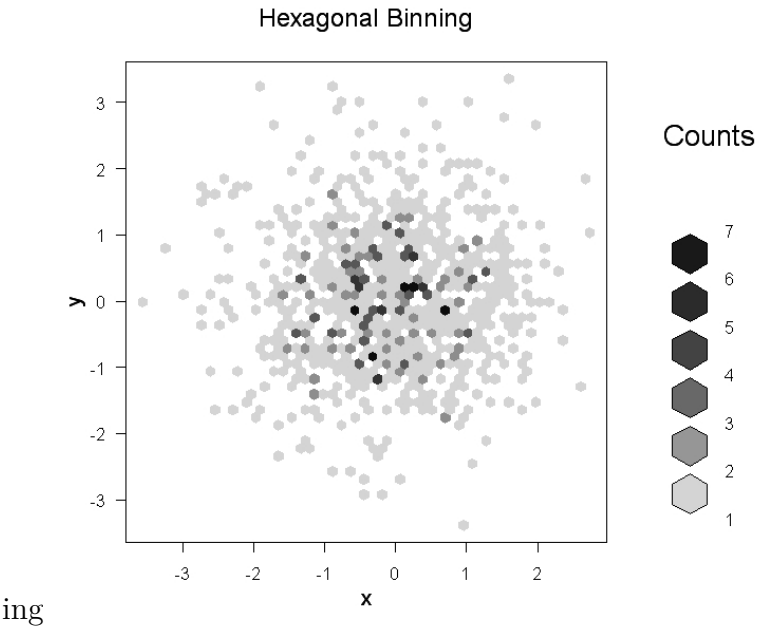
En ADVIZOR [ADV10] se utilizan tres técnicas para mejorar la escalabilidad. La primera de estas consiste en descartar puntos durante el *panning* y *zooming* para asegurar tiempos de respuesta interactivos, y que el usuario perciba un movimiento suave y continuo. Otra estrategia consiste en que si se codifica algún atributo en el color de los puntos, aquellos puntos más significativos se dibujen al final y de esta forman sean siempre visibles. Por último, se descartan los puntos superpuestos que igualmente no se verían.

Existen varias formas alternativas de incrementar la escalabilidad visual de los *scatterplots*, como lo son por ejemplo el *jittering* [Fri95], la representación de densidades de puntos y el *binning* [MM02; Car+87; Hao+10] (ver figura 2.6) En el capítulo 4 se presentarán varias modificaciones de los *scatterplots* que intentan superar sus factores limitantes.

Coordenadas paralelas

Los ParaBoxes [Eic00] (implementados en ADVIZOR [ADV10]) son la combinación de gráficos de coordenadas paralelas [ID90] con diagramas de cajas [Dal+95, Sec. 1.4] y gráficos de burbujas (ver figura 2.7). Dependiendo de los datos, pueden mostrar en el orden de 10^5 líneas y hasta cien columnas. El factor limitante de la escalabilidad visual de los ParaBoxes es el mismo que el de las coordenadas paralelas: la superposición causada por dibujar demasiadas líneas.

Zhao et al. [Zha+04] presentan la herramienta Visual Miner (V-Miner), usada por ingenieros de Motorola para identificar patrones significativos en los datos correspondientes a testeo y diseño de sus productos. Esta herramienta se basa principalmente en un gráfico de coordenadas paralelas complementado con lo que ellos llaman "figuras de tendencia". Estas figuras, una por cada eje paralelo (ver figura 2.8), refleja los valores que toma la secuencia de datos registrados para cada uno de los atributos. De esta forma se puede apreciar cómo fueron cambiando los valores a medida que se fueron generando o recolectando los datos.



ing

Figura 2.6: Scatterplot representado con *bins* hexagonales usando R[R], donde el color representa la densidad de puntos en cada celda. Figura tomada de `http://www.statmethods.net/graphs/scatterplot.html`.

Grafos y árboles

NicheWorks [Wil97] utiliza una serie de algoritmos que progresivamente van refinando el *layout*. Si el grafo es ralo o tiene una estructura regular que el algoritmo de posicionamiento pueda aprovechar, la escalabilidad tiende a ser considerablemente mayor. También utiliza símbolos simples que son fácilmente representables. Durante el *panning* y el *zooming*, los nodos y los enlaces menos significativos son dejados de lado para poder asegurar tiempos interactivos.

2.5.2. Aprovechando las interacciones

Agregarle interacciones a una técnica de visualización incrementa enormemente su potencial: estas permiten analizar la vista y encontrar más información sobre alguna porción de esta que parezca interesante. En este caso particular, nos interesa ver cómo las interacciones pueden mejorar la escalabilidad visual de una técnica.

Foco+contexto

El foco+contexto es una técnica que permite ampliar un área determinada de la visualización. En simultáneo, el área ampliada se muestra en contexto con el todo. Las vistas foco+contexto se basan en tres premisas planteadas por Card [Car03]:

1. El usuario necesita una visión general (contexto) e información detallada (foco).

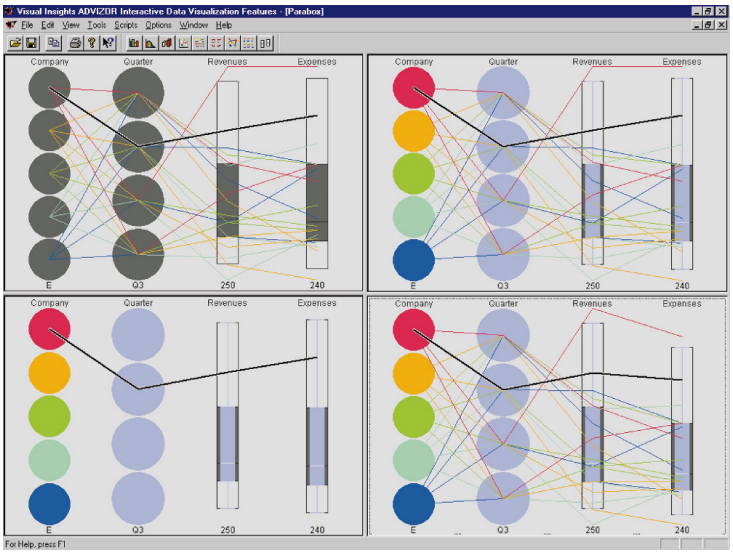


Figura 2.7: ParaBoxes implementados en ADVIZOR. Los ParaBoxes son la combinación de coordenadas paralelas, gráficos de cajas y de burbujas.

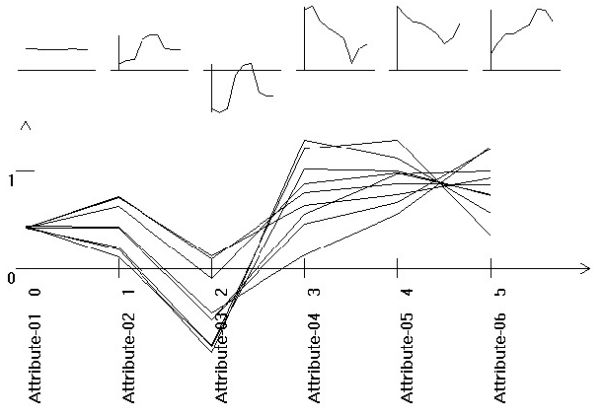


Figura 2.8: V-Miner. Gráfico de coordenadas paralelas mejorado con figuras de tendencia. De esta forma, independientemente de la cantidad de datos existentes, es posible ver cómo evolucionan los atributos a lo largo de los distintos registros.

2. La información necesaria en la vista general puede ser diferente de la necesaria en la vista detallada. La vista debe proveer información suficiente para que el usuario pueda continuar explorando la información o debe brindarle el contexto de la información detallada que se encuentra examinando.

3. La vista general y la detallada pueden combinarse dentro de una única visualización. Partir la información en varias visualizaciones tiende a degradar el rendimiento y demandar más tiempo por parte del usuario.

(a) *Zooming* es el incremento del tamaño de una fracción menor de una imagen 2D (o vice versa) bajo la restricción de que la ventana de visión sea de tamaño constante.

(b) *Panning* es el movimiento suave y continuo de la ventana de visión sobre una imagen 2D de un tamaño mayor.

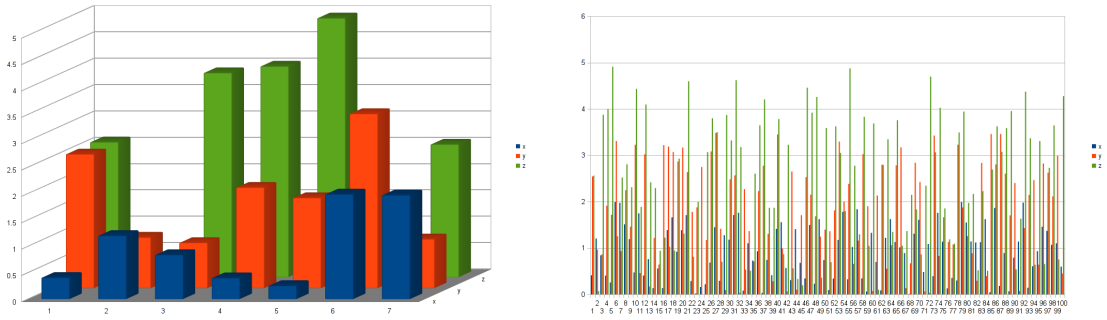
Figura 2.9: *Zooming* y *Panning* sobre una imagen 2D.

Zooming y *panning*

A diferencia del foco+contexto, que modifica la información que se visualiza, *zooming* y *panning* permiten modificar únicamente la vista, sin alterar la información que se muestra. En el caso de *zooming* (ver figura 2.9a) se puede alejar la vista para ver más datos, detectar algún área de interés y acercar para apreciar mejor los datos en esa área. En el caso de *panning* (ver figura 2.9b), se va desplazando el sector de la visualización que se muestra.

Metáforas multirresolución

Las metáforas multirresolución, reflejan diferentes niveles de detalle de los datos variando de una representación muy general con poco nivel de detalle a una representación más fina con alto nivel de detalle. Aplicable tanto a gráficos de barras, *scatterplots* o redes, consiste en mostrar los elementos visuales cada vez con menor detalle a medida que el observador se aleja (*zooming out*) de la visualización. Por ejemplo, como se muestra en la figura 2.10, un gráfico de barras en el mayor detalle puede mostrar solo algunas barras 3D, y a medida que disminuye el detalle se muestran más barras, pudiendo llegar a mostrar muchas barras de 1 único píxel de ancho o incluso barras superpuestas. En el caso de coordenadas paralelas, las coordenadas paralelas jerárquicas [Fua+99] se presentan como una estrategia para poder visualizar grandes conjuntos de datos, ofreciendo una vista multirresolución de los datos a través de *clustering* jerárquico.



(a) Pocos datos con el mayor detalle. **(b)** Muchos datos con menos detalle.

Figura 2.10: Ejemplo de diferentes resoluciones en un gráfico de barras. En la figura (a) se muestra un gráfico de barras con pocos datos y mucho detalle. Por otro lado, en la figura (b) se muestran muchos datos, pero con poco detalle en las barras que los representan.

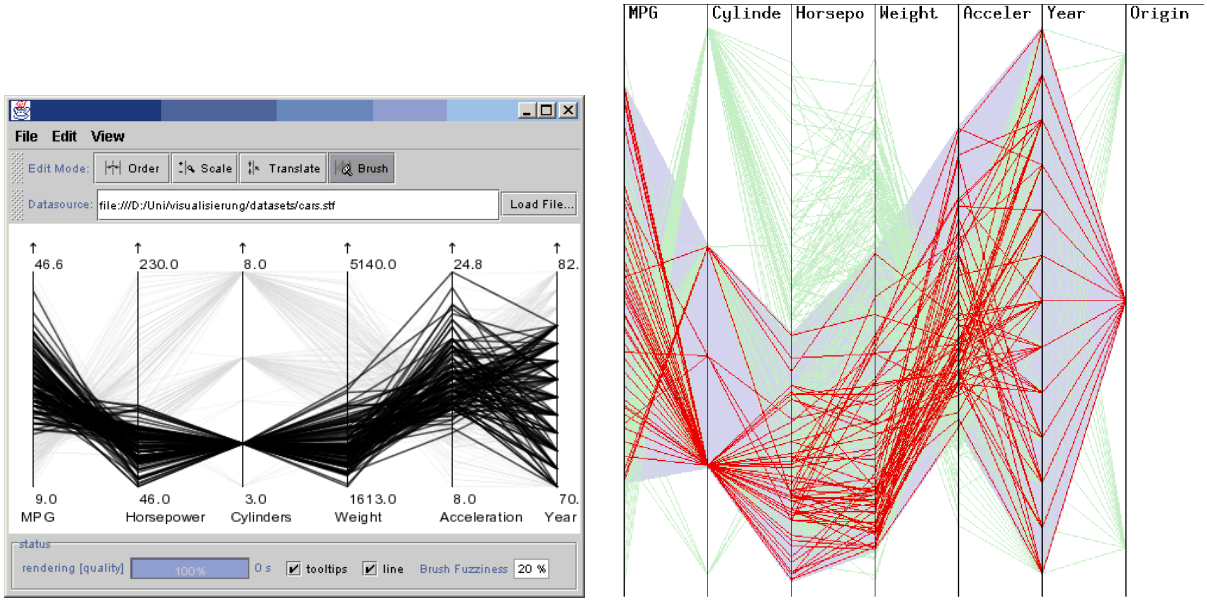
Selección

La selección implica filtrar y marcar interactivamente un subconjunto de los datos (por ejemplo, algunos nodos y enlaces en un grafo, o algunos datos en un gráfico de coordenadas paralelas) y *esconder* los ítems no seleccionados. La selección le permite a los usuarios incrementar la escalabilidad reduciendo la complejidad visual, y de esta manera concentrar la atención en los datos de interés. En la figura 2.11 se muestran ejemplos de selección correspondientes a Parvis [Hau+02] y a XmdvTool [MW95] en gráficos de coordenadas paralelas.

2.5.3. Enlazando vistas

Cada tipo de visualización tiene sus ventajas y sus desventajas. Las visualizaciones de grafos, por ejemplo, soportan datos altamente estructurados, pero no tienen gran escalabilidad visual; por otro lado, los gráficos de barras y los *scatterplots* son efectivos a la hora de lograr una vista resumida del conjunto de datos. Para aprovechar los beneficios complementarios de las distintas técnicas, es habitual generar múltiples vistas enlazadas del mismo conjunto de datos, cada vista con una técnica diferente.

De esta forma se logra complementar las metáforas visuales entre sí (reemplazando las desventajas de unas con las ventajas de otras) e incrementar la escalabilidad visual. XGobi [Swa+98] presenta vistas enlazadas de *scatterplots* y coordenadas paralelas (ver figura 2.12) y matrices de *scatterplots* enlazadas.



(a) Parvis **(b)** XmdvTool

Figura 2.11: Dos ejemplos de selección de datos en gráficos de coordenadas paralelas, correspondientes a las aplicaciones Parvis (a) y XmdvTool (b). En la figura (a) se han seleccionado todos los autos con 4 cilindros y en la figura (b), todos los autos europeos.

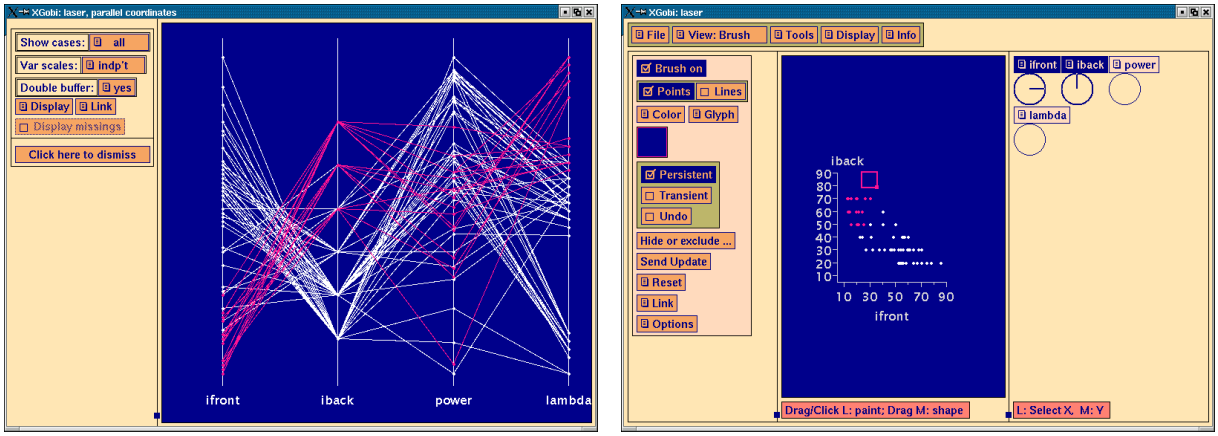


Figura 2.12: XGobi presenta la posibilidad de enlazar vistas: una usando gráficos de coordenadas paralelas y la otra *scatterplots*. En este ejemplo se ve cómo la selección de un subconjunto de los datos se aprecia en ambas vistas.

Capítulo 3

Métricas y predicción

Se introduce el concepto de métrica en el campo de la visualización, junto con el marco teórico definido por diferentes autores en este tema. Además, se presentan los diferentes estados y transiciones del MUV en el que se encuadra el aporte de esta tesis. Finalmente, se plantean los beneficios de la predicción en el marco de este modelo y la elección de la técnica de visualización adecuada para un dado conjunto de datos.

3.1. Introducción

El objetivo de una visualización es obtener una representación del conjunto de datos que ayude al usuario en la correcta interpretación de los mismos y así lograr un acertado análisis de estos. Dado el constante crecimiento de los conjuntos de datos en diferentes y variadas áreas de aplicación, la tarea de elegir la técnica más adecuada para visualizarlos convenientemente no es sencilla. Además, el resultado del proceso de visualización depende de todas las decisiones que se hayan tomado a lo largo de dicho proceso: un usuario inexperto es propenso a tomar decisiones equivocadas afectando negativamente la visualización obtenida y, a la larga, frustrando su experiencia con la visualización.

Dada la gran variedad de técnicas de visualización existentes es necesario contar con medidas que ayuden a determinar qué técnica es la más adecuada para un dado conjunto de datos.

En 1983 Tufte [Tuf01] busca medir la calidad de la visualización en función de la cantidad de tinta que consume y el propósito de esa tinta; sostiene que gran parte de la tinta en el gráfico debe representar información de los datos. La cantidad *data-ink* representa el núcleo no redundante y que no se puede eliminar del gráfico. Luego,

$$\text{Índice } \textit{data-ink} = \frac{\textit{data-ink}}{\text{total de tinta usada para realizar el gráfico}}$$

El índice *data-ink* representa la proporción de tinta del gráfico destinada a mostrar información no redundante de los datos y es equivalente a uno menos la proporción de tinta del gráfico que puede ser eliminada sin pérdida de información de los datos. Al diseñar

un gráfico se debe buscar, siempre dentro de lo razonable, maximizar el índice *data-ink*, y eliminar la información redundante y aquella que no represente información de los datos.

Miller et al. [Mil+97] también plantearon el interrogante sobre medir la "calidad" de una visualización:

> ¿Cómo podemos medir *cuán buena* es una visualización particular o la combinación de visualizaciones?

Para responder a esa pregunta, exploraron varios métodos de visualización temática de contenidos de grandes colecciones de documentos y concluyeron que es necesario contar con métricas que faciliten la comparación entre los diferentes métodos de visualización.

3.2. Marco de referencia

Varios autores se concentran en reafirmar la necesidad y utilidad de definir métricas para caracterizar el comportamiento de las diferentes visualizaciones. En particular, algunos presentan marcos de referencia en los cuales encuadrar las métricas definidas sobre las técnicas; entre estos pueden mencionarse algunos criterios para la evaluación de técnicas de visualización ([Fre+02]), una primera sistematización de las métricas de calidad ([BS06b]), guías para definir y comparar métricas de calidad ([Tat+10]) y un análisis de las métricas de calidad de visualizaciones de datos multi-dimensionales ([Ber+11]).

En particular, Freitas et al. [Fre+02] plantean la necesidad de contar con criterios para la evaluación de técnicas de visualización de información y definen cuatro clases de criterios para la evaluación de la usabilidad de una representación visual:

- *Completitud.* La completitud implica representar en la visualización todo el contenido semántico de los datos.
- *Organización espacial.* La organización espacial está relacionada con la diagramación de la representación visual, lo que comprende analizar cuán fácil es localizar un dato en la visualización y cuán consciente se es de la distribución de los datos en la representación.
- *Codificación de la información.* La codificación de la información incluye no solo el mapeo de datos a elementos visuales sino también el uso de símbolos adicionales o representaciones alternativas que ayuden en la percepción de la información.
- *Cambios de estado.* El tiempo que lleva reconstruir la visualización luego de una acción del usuario y los cambios en la organización espacial de la imagen resultante pueden afectar la percepción de la información.

También definen tres clases de criterios para la evaluación de las interacciones, que comprenden las tareas comunes realizadas por los usuarios:

- *Orientación y ayuda.* Se debe analizar, por ejemplo, el soporte al usuario para cambiar el nivel de detalle, rehacer/deshacer acciones o información adicional de utilidad.

- *Navegación y consultas.* Se debe analizar la posibilidad de seleccionar datos, cambiar el punto vista del usuario, manipular la representación geométrica de los elementos, buscar información específica y la posibilidad de mostrar elementos ocultos o en *clusters*.

- *Reducción del conjunto de datos.* Implica analizar la posibilidad de la técnica de reducir los datos a través del filtrado y la clusterización.

Bertini y Santucci [BS06b] analizan las métricas de calidad en términos generales y presentan una primera sistematización de estas. Proponen una clasificación basada en tres clases principales de métricas:

- *Métricas de tamaño.* Son medidas simples que sirven de base para otros cómputos y cuantifican características básicas de la visualización, por ejemplo, número de ítems, porcentaje de ocupación de la pantalla, etc.

- *Métricas de efectividad visual.* Se refiere a las métricas que miden la degradación de la imagen: colisiones, oclusión, etc.

- *Métricas de preservación de características.* Se refiere a las métricas que miden cuán correctamente una imagen representa alguna característica particular de los datos.

Tatu et al. [Tat+10] presentan una validación acerca de que las medidas de calidad pueden simular la selección de la "mejor" vista según la percepción humana; ellos validan la suposición de que los patrones capturados por las medidas se corresponden con los patrones capturados por el ojo humano. Para esto comparan medidas de calidad ya establecidas (consistencia de clases, densidad de clases e histogramas de densidad en 1D y 2D) a través de una evaluación empírica con test de usuarios.

Con el propósito de lograr proveer un marco común para estudiar las características de diversas métricas de calidad de visualizaciones de datos multi-dimensionales, Bertini et al. [Ber+11] presentan un análisis sistemático de las medidas publicadas en diversos trabajos. Caracterizan las medidas en función de diferentes factores en común:

- *Técnica.* Es la técnica de visualización a la cual se le aplica la medida.

- *Aspecto que mide.* Describe qué es lo que mide la métrica de calidad. Para el análisis se agruparon las métricas en 5 categorías: *clustering*, correlación, valores atípicos, calidad de la imagen y preservación de características.

- *Dónde se mide.* Considera si se mide sobre los datos, la imagen o ambos.

- *Propósito.* Describe la razón principal por la cual usar la métrica de calidad, es decir, el objetivo que persigue la métrica. Se identificaron 5 propósitos principales: proyección, ordenamiento, abstracción, mapeo visual y optimización de la vista.
- *Interacciones.* Se refiere a las interacciones que es posible realizar sobre las medidas, como por ejemplo seleccionar medidas diferentes o cambiar umbrales.

Con el fin de agilizar el proceso de visualización y garantizar una adecuada visualización de los datos se busca guiar al usuario en la selección de los diferentes parámetros involucrados en la visualización. Una guía adecuada depende, en parte, de la existencia de métricas que caractericen diferentes aspectos de las técnicas y ayuden a determinar qué técnica es la más adecuada para un dado conjunto de datos. No se pretende obtener un sistema que genere la visualización de forma totalmente automática, ya que eso sería una presentación, sino que el objetivo es contar con un sistema semiautomático que brinde al usuario una visualización que sea el punto de partida del proceso de exploración y análisis de los datos.

3.3. Guías para la definición de métricas

Algunos autores, además de presentar métricas específicas, también introducen guías para la definición de métricas. De esta forma se busca promover el estudio de la percepción de patrones visuales y su formalización en métricas computables. Por ejemplo, para dos de las clases de métricas definidas, Bertini y Santucci [BS06b] presentan el esquema de una metodología para definirlas:

- *Métricas de efectividad visual*
  1. Identificar un principio de optimización visual, que permita establecer que una configuración es “mejor” que otra.
  2. Definir una medida que cuantifique el criterio identificado anteriormente.
- *Métricas de preservación de características*
  1. Elegir una característica de interés.
  2. Definir formalmente dicha característica tanto en el espacio de los datos como en el de la visualización.
  3. Validar la definición visual con la percepción del usuario.
  4. Definir una métrica que compare los valores obtenidos del espacio de datos con los obtenidos en el espacio de la imagen.

Por otro lado, Tatu et al. [Tat+10] presentan guías para establecer un marco de referencia en la definición y comparación de métricas de calidad:

1. Seleccionar una técnica de visualización.

2. Seleccionar una característica visual.
3. Formalizar dicha característica a través de un algoritmo.
4. Llevar a cabo un test en el que los usuarios ordenen las imágenes en función de la característica elegida.
5. Analizar el resultado obtenido por los usuarios y comparar con las medidas calculadas.
6. Finalmente, refinar la métrica.

3.4. El Modelo Unificado de Visualización

Un modelo de visualización debe permitir a los usuarios concentrarse tanto en las operaciones como en los operandos, es decir tanto sobre las acciones necesarias para obtener cierto resultado como sobre las etapas que implica el proceso de obtención de dicho resultado. Si bien se han presentado diferentes modelos, estos han estado orientados a algún campo particular de la visualización ([CM97; CR98]).

El Modelo Unificado de Visualización ([Mar+03]) se presenta como *un modelo aplicable a cualquier visualización, independientemente del campo de origen de los datos*. En este modelo se ven representados los diferentes procesos que afectan al conjunto de datos (transformaciones) y las etapas por las que esos datos atraviesan (estados). Además, nos proporciona un modelo conceptual a partir del cual se pueden definir las interacciones de forma tal que quede bien establecido sobre qué conjunto de datos operan, cuál es el resultado que se obtiene y por lo tanto, cuál es el impacto general en el proceso.

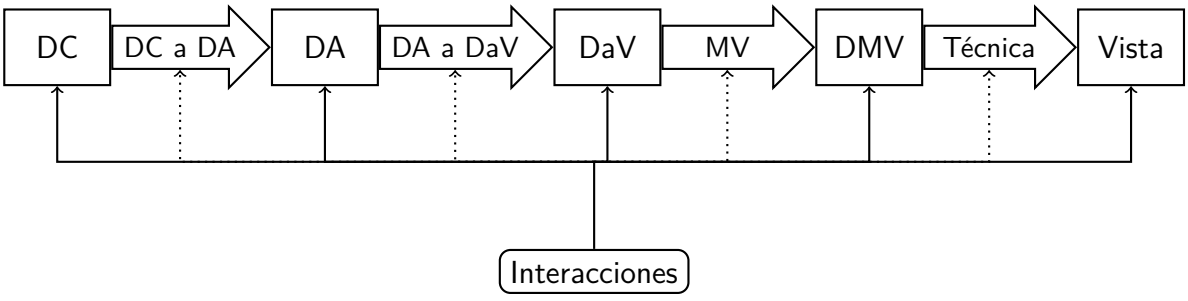


Figura 3.1: Pipeline del Modelo Unificado de Visualización (MUV)

A continuación se brinda una breve descripción de cada estado y de cada una de las transformaciones del Modelo Unificado de Visualización:

- *Estado: Datos Crudos (DC)* Es el conjunto inicial de datos. Son los datos recolectados directamente del dominio de aplicación. Pueden provenir de visualizaciones previas o de fuentes externas.

→ *Transformación: Datos Crudos a Datos Abstractos (DCaDA)* Permite al usuario seleccionar cuáles son los datos que quiere visualizar. Esta transformación se encarga de llevar los datos a la representación interna del sistema de visualización.

□ *Estado: Datos Abstractos (DA)* Son los datos seleccionados por el usuario para visualizar. No es condición que se visualicen todos los datos que se encuentran en este estado. Tampoco es condición que sea un subconjunto de los datos crudos, ya que los datos derivados de los datos crudos también son datos abstractos.

→ *Transformación: Datos Abstractos a Datos a Visualizar (DAaDAV)* Permite al usuario determinar exactamente *qué* datos estarán en la visualización. Esto datos deben ser un subconjunto de los datos abstractos.

□ *Estado: Datos a Visualizar (DAV)* Es el conjunto de datos que efectivamente se visualizarán.

→ *Transformación: Mapeo visual (MV)* El usuario determina *cómo* quiere visualizar los datos. En este proceso se define el sustrato espacial de los datos.

□ *Estado: Datos Mapeados Visualmente (DMV)* Es el conjunto de datos con toda la información necesaria para visualizarlos mediante alguna técnica que los soporte, es decir, son los datos con una estructura visual y un sustrato espacial asociado.

El *sustrato espacial* especifica cómo estará organizado el espacio. Esto es de suma importancia ya que la ubicación espacial de los elementos es *perceptual*mente dominante.

Los *elementos visuales* y sus *atributos gráficos* asocian atributos geométricos a los datos que se representarán en pantalla. En este estado se determinan características generales de la representación que se desea. El máximo grado de detalle queda resuelto en la etapa siguiente, en la cual se elige la técnica.

La *estructura visual* está definida por el sustrato espacial, los elementos visuales y los atributos gráficos de los elementos visuales.

→ *Transformación: de Visualización (Técnica)* Es la aplicación de una técnica que soporte el mapeo visual definido en la etapa previa. Además, determina todos los demás elementos que componen la escena, tales como colores, luces, etc. Agrega elementos que son parte de la escena pero son adicionales a la visualización de los datos.

□ *Estado: Datos Visualizados (Vista)* Esta es la representación final de los datos a visualizar con la técnica elegida. Es el punto de acceso del usuario al Pipeline de Visualización. En esta etapa el usuario interactúa con la visualización en el proceso de exploración y análisis de los datos.

A pesar de que el usuario interactúa sobre esta última etapa, la *Vista*, sólo algunas interacciones se resuelven allí. En general, resolver una interacción implica modificar una o varias etapas del Pipeline de Visualización.

La *Vista* es el punto de entrada para todas las interacciones, tanto para las que pueden resolverse en la propia vista como para aquellas que en realidad se resuelven en estados o transformaciones anteriores. A pesar de que los efectos de las interacciones se reflejan en la *Vista*, estas pueden afectar cualquier estado o trasformación del Pipeline de Visualización (ver figura 3.1).

3.5. Un sistema de visualización asistido por medidas

La calidad de una visualización se podría medir a lo largo de diferentes etapas del Pipeline de Visualización, siendo la *Vista* la etapa más directa para evaluar el resultado. Sin embargo, realizar una evaluación de la visualización a esta altura del proceso implica generar la visualización aunque esta no vaya a resultar efectiva. Nuestro objetivo es *predecir* la calidad de una visualización antes de alcanzar la *Vista*, es decir antes de aplicar la técnica al conjunto de datos. La etapa en la que es posible realizar algún tipo de *predicción* es la *transformación de vista* dado que en este punto ya se conoce el conjunto de datos a visualizar y el paso siguiente es elegir la técnica con la que se realizará la visualización. En esta transformación, y antes de aplicar la técnica para obtener la *Vista*, es posible calcular medidas propias de cada técnica que predigan el resultado de la aplicación de dicha técnica al conjunto de datos. De esta forma, cada técnica debe estar acompañada de un conjunto de medidas que permitan predecir, para un dado conjunto de datos, cómo se comporta en función de sus factores limitantes.

En este contexto, contar con una medida que modele algún aspecto de la escalabilidad visual de una técnica (como un parámetro de la calidad de una visualización) es uno de los elementos a tener en cuenta a la hora de diseñar la visualización de un determinado conjunto de datos.

En el proceso de visualización las medidas pueden utilizarse en diferentes etapas, tanto en la elección como en la configuración de la técnica. Llegando al final del proceso, una vez elegida la técnica de visualización y los parámetros de esta (tamaños, distancias, etc.), el usuario puede decidir, a partir del valor resultante de la evaluación de las métricas asociadas a la técnica, si la visualización sería aceptable o no para su propósito. Si el valor es aceptable, entonces se puede proceder con la realización de la visualización. En cambio, si el valor no es aceptable, se debe comenzar con el proceso de refinamiento de los parámetros de la técnica (ver figura 3.2).

En una etapa anterior, las métricas se pueden utilizar para guiar la elección de la técnica. El objetivo es evitar que el usuario intente visualizar conjuntos de datos con técnicas que, independientemente de la configuración, no darán buenos resultados pero, en cambio, sí intente con aquellas que puedan resultar en visualizaciones *potencialmente* aceptables. Esto se puede lograr, una vez elegida la técnica y antes de comenzar con el refinamiento de sus parámetros, calculando los valores límites de la métrica, es decir, llevando los parámetros que influyen en el cálculo de las medidas a los extremos que maximicen (o minimicen) su valor. Si estos valores límites no son aceptables, la técnica

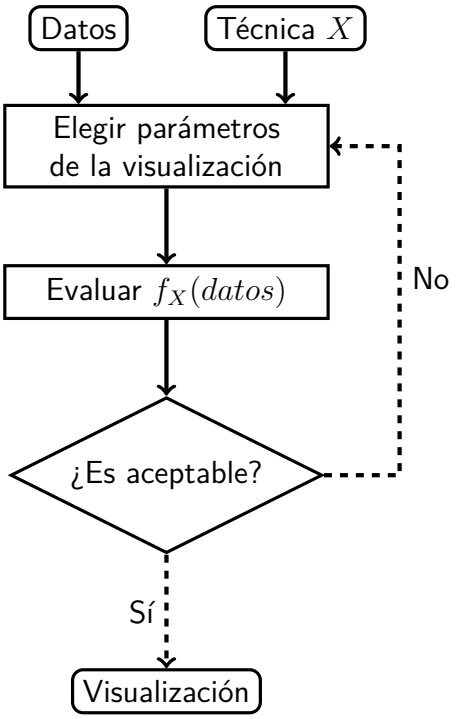


Figura 3.2: Proceso de refinamiento de los parámetros de la técnica. Las transiciones marcadas con una línea discontinua son aquellas que deberían contar con asistencia del usuario.

se puede descartar para ese conjunto de datos, por ejemplo, en un *scatterplot* tradicional, la superposición puede ser inaceptable aún cuando el tamaño de la visualización sea del tamaño de la pantalla y los glifos tengan un píxel de tamaño. Sin embargo, si los valores límites son aceptables indica que la técnica es *potencialmente* aceptable y, refinando los parámetros de esta, se puede alcanzar una buena visualización. En la figura 3.3 se muestra este proceso.

Finalmente, y generalizando el caso anterior, las métricas también se pueden utilizar para determinar cuál es el conjunto de técnicas que resultarían en visualizaciones *potencialmente* aceptables para un dado conjunto de datos. De esta forma, cuando el usuario tiene que elegir una técnica para visualizar su conjunto de datos, no necesita elegir una del total de técnicas disponibles sino solamente de un subconjunto selecto (ver figura 3.4).

La definición de métricas asociadas a las técnicas es un elemento de gran utilidad al momento de determinar la técnica más apropiada para visualizar un determinado conjunto de datos. Sin embargo, las métricas por sí solas no son suficientes ya que estas no tienen en cuenta el tipo de dato que se está visualizando. Por lo tanto, cada vez que se propone evaluar el comportamiento de una técnica con un conjunto de datos, se asume que existió una instancia previa (por ejemplo, usando la semántica de los datos [Lar10; Esc+11]) en la cual se identificó esa técnica como apta para ese tipo de datos.

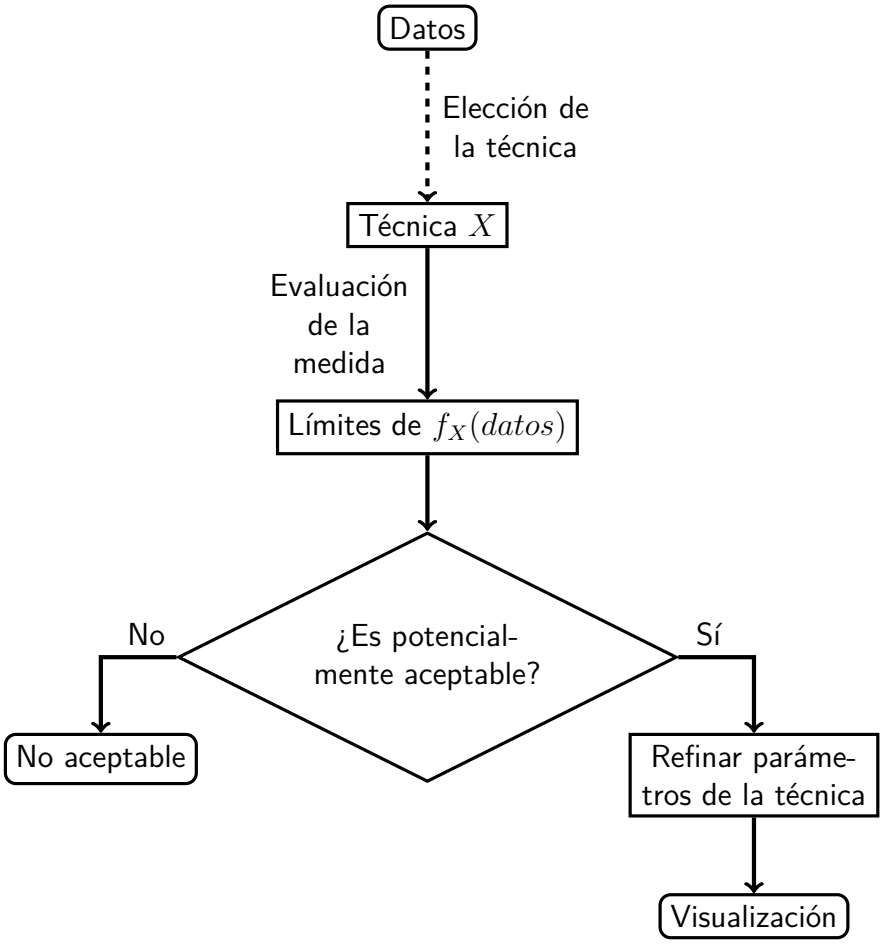


Figura 3.3: Proceso de elección de los parámetros de la técnica de visualización seleccionada teniendo en cuenta las métricas definidas.

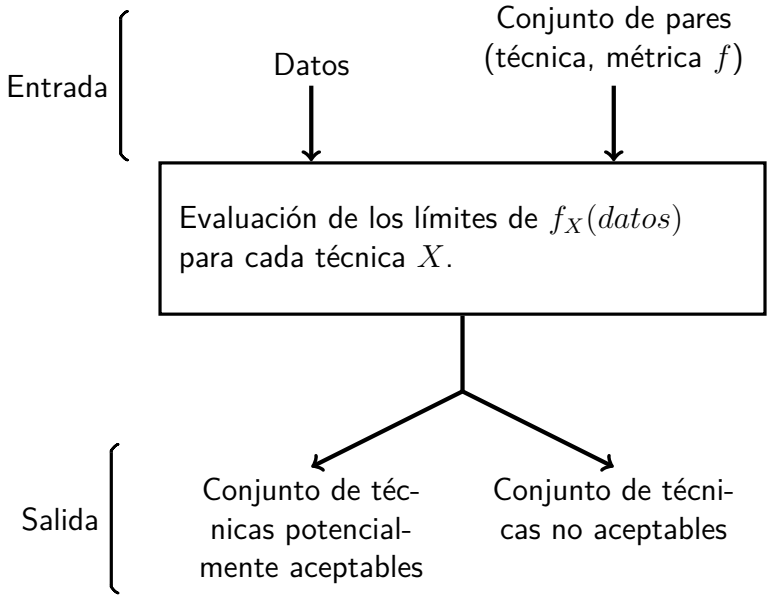


Figura 3.4: Dado el conjunto de datos que se desea visualizar, selección de las técnicas cuyo valor de las métricas asociadas *predicen* una visualización potencialmente aceptable.

Capítulo 4

Scatterplots

Se presenta el estado del arte de la técnica de scatterplots que se ha realizado teniendo en cuenta la evolución de dicha técnica para representar distintos conjuntos de datos. Los scatterplots son gráficos 2D útiles para representar grandes conjuntos de datos; originalmente se han usado para representar conjuntos de datos escalares bidimensionales y posteriormente fueron adaptados para representar datos multidimensionales. Para esto se han presentado distintas variantes que amplían y mejoran la técnica teniendo en cuenta, en cada caso, los puntos débiles correspondientes a cada representación.

4.1. Introducción

En *MathWorld* se define *scatterplot* [RW] como:

> Un *scatterplot* (o diagrama de puntos) es una visualización de la relación entre dos variables medidas a un mismo conjunto de individuos.

Esta visualización es un diagrama que usa coordenadas cartesianas para mostrar los valores de un conjunto de datos bidimensional con un punto ($\cdot$). La posición de cada punto en el diagrama está determinada por los dos atributos del dato que se está representando. Es posible complementar el diagrama con información adicional, tal como ejes, etiquetas, leyendas o títulos y también con líneas de regresión o curvas suaves.

4.2. Limitaciones de los *scatterplots*

Los *scatterplots* tienen dos grandes limitaciones: la cantidad de dimensiones representables y la cantidad de datos que se pueden visualizar manteniendo una gráfica significativa a pesar de la superposición existente.

4.2.1. Dimensionalidad

Si bien la técnica es inherentemente bidimensional, el concepto de *scatterplot* puede extenderse a un diagrama tridimensional [Don+88], en el que los puntos se ubican en po-

siciones xyz del espacio. En ambos casos, es posible representar datos multidimensionales reemplazando los puntos por *glifos*. Un glifo es un símbolo que tiene tamaño, forma y color. Del total de dimensiones de los datos, se elijen dos (o tres, dependiendo del caso) para ser representadas en los ejes coordenados y las restantes se mapean a los atributos del glifo (diferentes tamaños, formas y/o colores). Los glifos pueden variar desde letras o formas geométricas simples ($\square$, $\odot$, $*$) hasta figuras más complejas, como por ejemplo las caras de Chernoff [Che73] o las familias de íconos como la presentada por Pickett y Grinstein [PG88].

Otra alternativa para representar datos multidimensionales es la llamada *matriz de scatterplots*. Esta matriz cuenta con diferentes *scatterplots* bidimensionales formados tomando las dimensiones de a pares y tiene la propiedad de que cualquier par adyacente de gráficos tiene un eje en común.

Ambas alternativas para representar datos multidimensionales, tienen limitaciones en cuanto a la cantidad de dimensiones representables. En el caso de la matriz de *scatterplots*, conviene evitar que los gráficos sean muy pequeños, por lo tanto, un número razonable sería hasta aproximadamente 10 dimensiones. Por otro lado, utilizando glifos complejos es posible representar una gran cantidad de dimensiones (con las Caras de Chernoff [Che73], por ejemplo, se puede representar hasta un máximo de 18 atributos). Sin embargo, un glifo muy complejo implica que debe ser de un tamaño considerable para apreciar los detalles y distinguir un dato de otro; esto disminuye la cantidad de datos que se pueden visualizar. Borgo et al.[Bor+13] presentan un estado del arte de las visualizaciones basadas en glifos.

4.2.2. Superposición

Cuando el conjunto de datos es grande, los *scatterplots* tienden a presentar un alto grado de solapamiento entre los glifos. Dependiendo de la aplicación particular de la visualización se puede estar frente a dos posibles escenarios:

1. es necesario distinguir los glifos entre sí, o

2. alcanza con identificar la densidad de glifos en las diferentes zonas del *scatterplot*.

En ambos casos, el solapamiento es un problema que dificulta el análisis de la visualización.

Para el primer caso, se plantean soluciones tales como diferentes glifos, distorsión y multirresolución. En el caso de necesitar distinguir diferentes densidades, se utilizan técnicas como transparencias, histogramas, *binning*, etc.

4.3. Extensiones de los *scatterplots*

En esta sección se realiza una recopilación de las técnicas más importantes basadas en *scatterplots* y las soluciones planteadas en cada caso para obtener visualizaciones exitosas más allá de sus limitaciones. Para una revisión histórica de los comienzos de los *scatterplots* se recomienda leer Friendly y Denis [FD05]. En la tabla 4.1 se detalla qué problema intenta

resolver cada trabajo y mediante qué estrategia. El estado del arte se realiza teniendo en cuenta el desarrollo de las técnicas de acuerdo a su presentación cronológica, ya que muchas estrategias toman ideas de publicaciones anteriores. Las figuras a las que se hace referencia se muestran al final del capítulo.

Año	Sección	Referencia	Limitación que soluciona	Estrategia
1973	4.3.1	[Che73]	Dimensionalidad	glifos complejos
1984	4.3.2	[CM84]	Superposición	glifos
1987	4.3.3	[BC87]	Dimensionalidad	matriz de scatterpltos
1987	4.3.4	[Car+87]	Densidad	*binning*
1988	4.3.5	[PG88]	Dimensionalidad	glifos
1997	4.3.6	[Bec97]	Densidad	3D, transparencias
2003	4.3.7	[WF03]	Superposición	multirresolución
2004	4.3.8	[Pir+04]	Superposición, densidad	vistas enlazadas, 3D, histogramas, color
2009	4.3.9	[Wic09]	Superposición, densidad	transparencia, distorsión, *binning*, etc.
2010	4.3.10	[Kei+10]	Superposición	distorsión

Dimensionalidad El *scatterplot* tradicional está limitado a 2 dimensiones. Se han propuesto diversas extensiones de la técnica para solucionar este problema y poder representar datos complejos: *scatterplots* 3D, glifos y matrices de *scatterplots*.

Superposición El mayor problema de los *scatterplots* es el alto nivel de solapamiento cuando el conjunto de datos visualizado es grande, ya que puede ocultar una parte significativa de los datos mostrados.

Estimación de densidades La inherente superposición de datos acarrea el problema de estimar la densidad de datos en cierta posición del gráfico. Los *scatterplots* representan correctamente la densidad sólo si no hay datos que se proyecten en los mismos píxeles, ya sea por la resolución limitada de la pantalla o porque hay varios datos iguales en el conjunto representado.

Tabla 4.1: Técnicas basadas en *scatterplots* presentadas en esta sección y descripción de las limitaciones que intentan solucionar.

4.3.1. Caras de Chernoff (1973)

Dado que los humanos pueden discriminar bastante bien entre caras muy similares, las caras de Chernoff [Che73] son un glifo especial que asocia variables a características faciales. Cada punto en un espacio $k-$dimensional ($k \leq 18$) se representa con la caricatura de una cara, cuyas características, tales como el tamaño y la forma de los ojos, la nariz, la boca, las pupilas, las cejas y el contorno de la cara, se corresponden con las componentes del punto.

En el trabajo original, Chernoff presenta dos ejemplos. Uno es una muestra de 87 especímenes fósiles con 8 variables medidas a cada uno (ver figura 4.1a) y el otro corresponde al análisis mineral de 53 muestras geológicas con 12 variables para cada muestra (ver figura 4.1b). En la tabla 4.2 se presenta una breve descripción de qué características de la cara se pueden mapear a cada uno de los parámetros (en el trabajo se plantean un máximo de 18 atributos).

4.3.2. Sunflowers (1984)

Cleveland y McGill [CM84] presentan las *sunflowers* como una estrategia para solucionar el problema de superposición en *scatterplots*. La región para graficar se divide en celdas cuadradas de igual tamaño. Se cuenta la cantidad de puntos que caen en cada celda y esa cantidad se representa con símbolos llamados *sunflowers*. Un punto se corresponde con cantidad 1, un punto con 2 segmentos se corresponde con cantidad 2, un punto con tres segmentos se corresponde con cantidad 3, y así siguiendo. La figura 4.2 muestra cómo los *sunflowers* evidencian la existencia de dos *clusters* de datos que no se notan en un *scatterplot* tradicional. El conjunto de datos fue generado a partir de tres distribuciones (una uniforme y dos binormales con centros en $(0; 0)$ y en $(4; 4)$).

4.3.3. Brushing (1987)

Con el objeto de facilitar el análisis de datos Becker y Cleveland [BC87] introducen la idea de seleccionar o "pintar" (*brushing*) datos en una matriz de *scatterplots*. En todos los casos el pintado puede ser transitorio, persistente o revertido. Sobre una matriz de *scatterplots* se permite, interactivamente:

- *Resaltar.* Los datos pintados en algún *scatterplot* se resaltan en todos los *scatterplots* de la matriz.
- *Borrar.* Los datos pintados se borran de todos los *scatterplots* de la matriz.
- *Shadow.* Los datos pintados se resaltan mientras que los no pintados se ocultan de todos los *scatterplots* de la matriz exceptuando del activo.
- *Etiquetar.* Si cada dato tiene un nombre asociado, el pintado permite mostrar el nombre en todos los *scatterplots* de la matriz. Para evitar solapamiento no es posible mostrar todas las etiquetas simultáneamente y con esta interacción se muestran bajo demanda.

En la figura 4.3 se muestran ejemplos de las operaciones que se definieron en el artículo.

4.3.4. Representación de densidades (1987)

Carr et al. [Car+87] plantean que "la clave para mostrar grandes conjuntos de datos es estimar y mostrar densidades." En la sección 2.2 ya se ha hablado de qué se considera un gran conjunto de datos; esto no depende únicamente de la cantidad y las características de datos, sino también del contexto en que estos se visualizarán.

En una matriz de *scatterplots* la selección de la estrategia de representación de densidades está sujeta a diversos factores siendo estos la interpretación no ambigua, el impacto visual directo, la facilidad de comparación, una buena resolución para las densidades, una

buena resolución para el posicionamiento, la alta velocidad de reexpresión[1], la adecuación para la selección de subconjuntos y la adecuación para gráfico pequeños. En el artículo prefieren las escalas de grises y métodos equivalentes para representar densidades ya que la resolución de posicionamiento es alta (cada píxel puede representar una densidad diferente), la velocidad de reexpresión es en tiempo real y los colores delimitan contornos (que tienen una interpretación no ambigua).

Además de la elección de color, se elije un símbolo para representar densidades: el hexágono. Basándose en las *sunflowers* [CM84] que clusterizan los datos dividiendo el espacio en celdas cuadradas, en este trabajo se divide el espacio en hexágonos, aludiendo a no enfatizar las direcciones vertical y horizontal. En la figura 4.4 se compraran los mismos datos con la división cuadrada y la hexagonal del espacio visualizando con *sunflowers*. En las figuras 4.5 y 4.6 se muestra la división hexagonal del espacio combinada con una escala de colores y con la variación de tamaño de los hexágonos, respectivamente.

4.3.5. Familias de íconos (1988)

Cuando los íconos se muestran en conjunto en un *scatterplot*, la estructura estadística de los datos se percibe en forma de contornos o gradientes en la textura que se forma a partir de la visualización.

El ejemplo presentado por Pickett y Grinstein [PG88] muestra cómo usar una familia de íconos para visualizar los datos de imágenes multiespectro. La familia de íconos diseñada para la visualización tiene 12 íconos (ver figura 4.7) y cada uno consiste de 5 segmentos conectados. Un segmento es el cuerpo del ícono y los restantes son las extremidades. Esta familia de íconos se utiliza para visualizar imágenes satelitales del clima (ver figura 4.8). Estas imágenes son multiespectro, tienen 5 canales y cada canal controla la orientación de las extremidades del ícono.

Si bien con esta familia de íconos en particular es posible visualizar hasta 5 atributos numéricos y grandes cantidades de datos, la complejidad de esta estrategia está en encontrar una familia de íconos adecuada y mapear acertadamente los atributos en las características del ícono.

4.3.6. *Vóxels* y transparencias en *scatterplots* 3D (1997)

Becker [Bec97] presenta un método para tratar de solucionar las dificultades que surgen de visualizar con *scatterplots* 3D tanto grandes cantidades de puntos como variables

[1] El término *reexpresión* fue acuñado, aunque no definido, por Tukey [Tuk77]. Si bien Carr et al. [Car+87] tampoco lo definen, otros autores han dado sus propias definiciones del término. Para DiBiase et al. [DiB+92] el término denota representaciones gráficas alternativas cuya estructura se alteró a través de transformaciones sobre los datos originales. Un ejemplo simple es transformar medidas en escalas lineales a logarítmicas o aplicar filtros de suavizado. Por otro lado, para Ratner [Rat11] el término significa cambiar la composición, estructura o escala original de una variable aplicando funciones (aritméticas, matemáticas o de truncado) para producir expresiones alternativas de la variable original.

categorizadas y valores desconocidos. A grandes rasgos, la técnica implica *voxelizar* utilizando celdas y *renderizar* el volumen resultante.

La técnica consiste en utilizar celdas uniformes para *discretizar* las variables reales mapeadas a los ejes, y donde los distintos valores obtenidos definen las celdas en los ejes categorizados. Esto divide al espacio en celdas volumétricas (*voxels*). El balance entre exactitud y velocidad de *rendering* se determina con la resolución de la celda. La opacidad de cada *voxel* es función de la cantidad de datos incidentes en él y el color se deriva promediando los valores de algún atributo de los datos incidentes. Los demás atributos de los datos se mapean a *sliders*.

De existir valores desconocidos (NULL), se tratan de manera especial; en este caso $\text{NULL} + v1 + v2 = \text{NULL}$ y $\text{promedio}(\text{NULL}, v1, v2) = \text{promedio}(v1, v2)$. A los propósitos de la visualización y para evitar la confusión con otros valores, los valores NULL se ubican debajo del origen para la coordenada correspondiente. En la figura 4.9 se muestra un ejemplo con 5 atributos, uno de los cuales se mapea a la opacidad, otro al color y los tres restantes a los ejes x, y y z.

4.3.7. Caricaturas Gestalt (2003)

Basándose en la premisa que la estructura visual de una imagen debe corresponderse con la estructura de los datos que intenta transmitir, Wattenberg y Fisher [WF03] proponen un método multirresolución para evaluar la organización *perceptual* de gráficos con información. Si bien el método que proponen es general, muestran cómo es posible aplicarlo a visualizaciones con *scatterplots* para obtener multirresolución.

En el trabajo simplifican el dominio de aplicación eliminando colores, profundidad, movimiento e interacción, y se enfocan únicamente en imágenes 2D no interactivas, en escalas de grises sin animación. Los principales pasos del modelo son:

1. Usando la teoría de espacio–escala generan varias escalas de la imagen.

   La teoría de espacio–escala busca representar imágenes en múltiples escalas, representando la imagen como una familia de imágenes suavizadas, parametrizadas por el tamaño del *kernel* de suavizado usado para eliminar los detalles.

2. Luego, detectan bordes cerrados en cada una de las diferentes escalas y pintan las áreas delimitadas con un gris promedio. A esta técnica la llaman *caricaturas Gestalt*.

3. Finalmente, se enlazan las características a diferentes escalas, para generar la estructura jerárquica correspondiente a la imagen.

En las figura 4.10 se ilustran estos pasos y en la figura 4.11 se ilustra la aplicación en *scatterplots* usando como ejemplo el diagrama astronómico de Hertzprung-Russell [SG93].

4.3.8. Composición de *scatterplots* 2D y 3D (2004)

Piringer et al. [Pir+04] proponen una combinación de *scatterplots* 2D y 3D, para solucionar el problema de la superposición. Enlazando las dos vistas es posible interactuar en 2D y obtener realimentación en 3D. Esa realimentación es enriquecida con información de profundidad (color y tamaño del punto) y con histogramas en 2D y 3D que muestran información adicional sobre la densidad de puntos.

Para codificar la información de profundidad varían el tamaño del punto y el color. Esto genera que en zonas superpobladas por puntos casi del mismo color sea imposible distinguir uno de otro. La solución que proponen es delinear la forma de cada punto dibujando un halo alrededor (un círculo fino semitransparente del mismo tono del punto pero con mucho menos brillo). Aunque esta solución contribuye a incrementar la superposición, plantean que no es preocupante dado que lo único visible hasta el momento era un conglomerado de nodos indistinguibles.

Con respecto al color para codificar la profundidad, basándose en las características del sistema visual humano [Tou97], elijen un mapa de colores de tal forma que los colores brillantes y cálidos (rojo o amarillo) sean para puntos próximos al observador y los colores oscuros y fríos (azul o gris) para puntos lejanos.

Para solucionar la falta de información de densidad de puntos, incorporan histogramas 2D en cada cara del cubo donde el *scatterplot* 3D es dibujado. Para orientar el histograma proponen dos alternativas (ver figura 4.12): prismas alineados a los ejes coordenados y texturas.

El zoom lo representan mostrando sólo una porción cúbica de los datos, y muestran el contexto espacial de forma similar a las densidades: todos los puntos que quedan fuera del cubo de visión y que son parte del contexto, se proyectan desde el centro sobre la superficie del cubo y se agrupan de acuerdo a la función deseada (ver figura 4.13). Luego, al resultado se le aplica un escalado lineal o logarítmico y se muestra a través de la opacidad de texturas o prismas.

Esta composición de técnicas fue utilizada para analizar datos de simulaciones de flujo en ingeniería automotriz (aproximadamente 32 000 celdas, 18 atributos, y 100 estampillas de tiempo).

4.3.9. `ggplot2` (2009)

`ggplot2` [Wic09] es un paquete de R para generar gráficos estadísticos o de datos. Está basado en la gramática definida en *Grammar of Graphics* [Wil05] que cuenta con un conjunto independiente de componentes a partir de los cuales se pueden crear nuevos gráficos, sin limitarse a un conjunto predefinido de implementaciones. En este contexto, `ggplot2` brinda herramientas para implementar varias soluciones a la superposición de datos en un *scatterplot*:

- la superposición leve puede solucionarse con puntos más pequeños o con glifos huecos. Ver figura 4.14.

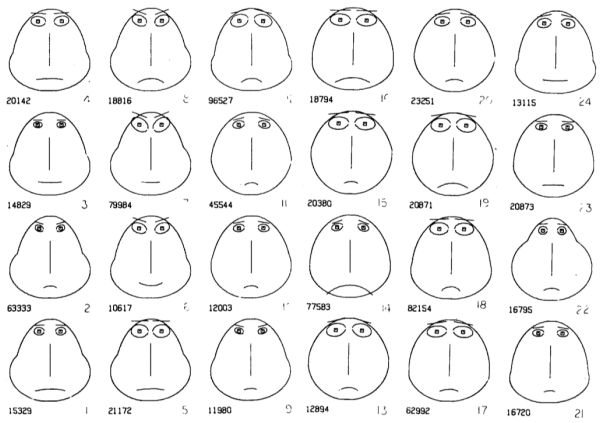
- para grandes conjuntos de datos con mucha superposición se pueden utilizar transparencias.
- en el caso de datos discretos, o datos que tiendan a ser discretos, se pueden combinar transparencias con *jitterring* [Fri95], que implica agregar un pequeño ruido a la posición de cada punto. Ver figura 4.15.
- crear celdas de puntos y contar la cantidad de puntos en cada celda (corresponde a un histograma en dos dimensiones). Se pueden visualizar con cuadrados o con hexágonos (a través del paquete `hexbin`). Ver figura 4.16.
- estimar la densidad en dos dimensiones con el paquete `stat_density2d` y dibujar en el *scatterplot* los contornos de las densidades, o mostrar las densidades con diferentes colores o tamaños de puntos. Ver figura 4.17.
- agregar al *scatterplot* resúmenes de los datos que muestren tendencias, por ejemplo una línea mostrando la media de los datos en algún eje.

4.3.10. *Scatterplots* generalizados (2010)

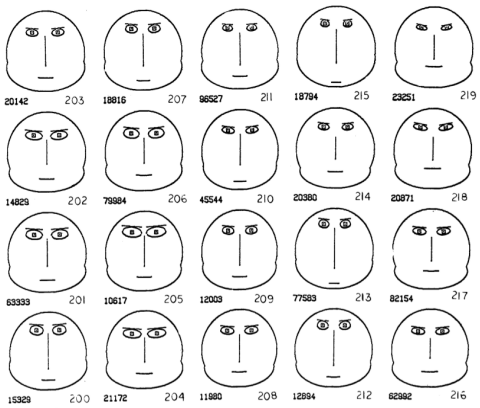
Keim et al. [Kei+10] proponen generalizar los *scatterplots* para permitir una representación de grandes conjuntos de datos que quepa en la pantalla y sea libre de superposición. La idea es permitirle al analista variar el grado de solapamiento y distorsión para generar la mejor vista posible.

Este *scatterplot* con superposición optimizada permite variar el grado de distorsión y el grado de solapamiento. La distorsión otorga más espacio a las áreas con alta densidad y menos espacio a las áreas con baja densidad de puntos, manteniendo a la vez la relación de vecindad entre los puntos y una igual distribución de las dimensiones x e y.

Se puede ajustar entre 0 % y 100 % tanto el nivel de distorsión como la optimización del solapamiento, donde 0 % de optimización de superposición corresponde al nivel de superposición inducido por los datos y 100 % se corresponde con la completa eliminación de la superposición, si es posible para el tamaño de ventana dado. En la figura 4.18 se puede ver cómo se distorsiona la visualización para diferentes valores de ambos parámetros.



(a) Caras de Chernoff correspondientes a 24 de las 87 muestras de especímenes fósiles representando 8 atributos cada una. (Página 37)

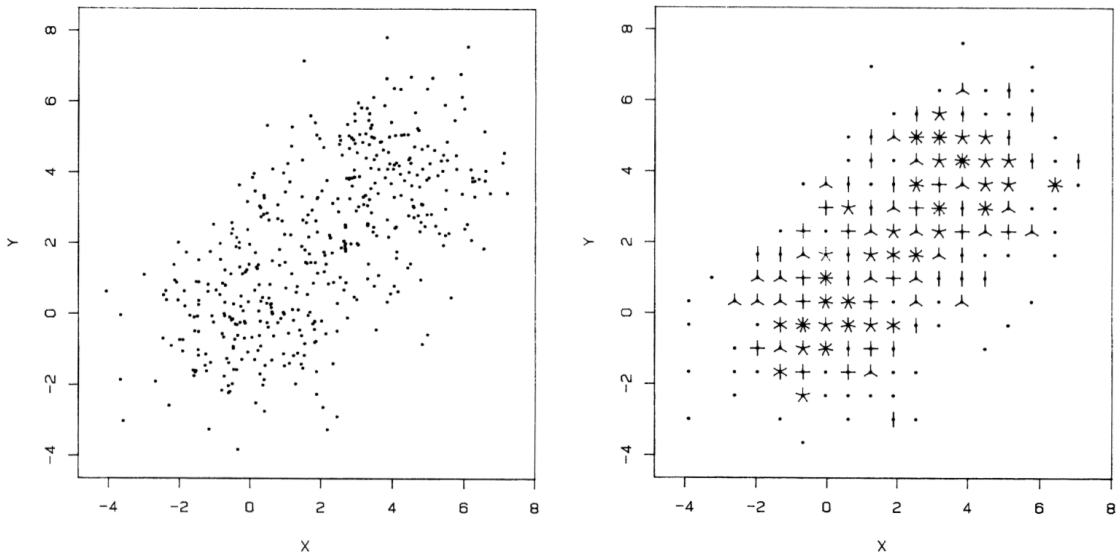


(b) Caras de Chernoff correspondientes a 20 de las 53 muestras geológicas representando 12 atributos cada una. (Página 37)

Figura 4.1: Dos conjuntos de datos representados con caras de Chernoff.

	Brief descriptive phrase
x_1	radius to corner of face, |OP|
x_2	angle of OP to horizontal
x_3	vertical size of face, |OU|
x_4	eccentricity of upper face
x_5	eccentricity of lower face
x_6	length of nose
x_7	vertical position of mouth
x_8	curvature of mouth
x_9	width of mouth
x_{10}	vertical position of eyes
x_{11}	separation of eyes
x_{12}	slant of eyes
x_{13}	eccentricity of eyes
x_{14}	size of eyes
x_{15}	position of pupils
x_{16}	vertical position of eyebrows
x_{17}	slant of eyebrows
x_{18}	size of eyebrows

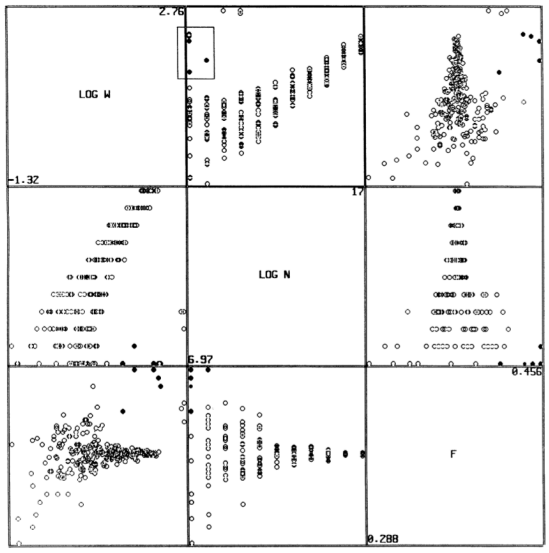
Tabla 4.2: Tabla que presenta las características faciales que se corresponden con los 18 atributos posibles de representar con las caras de Chernoff. (Página 37)



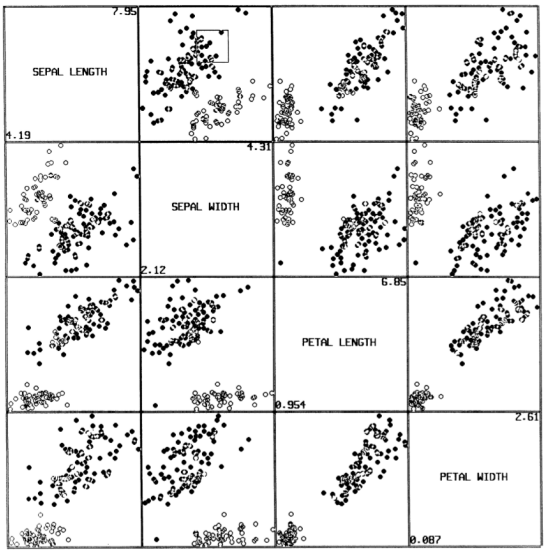
(a) Visualización con un *scatterplot* tradicional. Los datos aparentan formar un único *cluster* de puntos.

(b) Visualización con *sunflowers*. Se evidencian dos *clusters* de puntos.

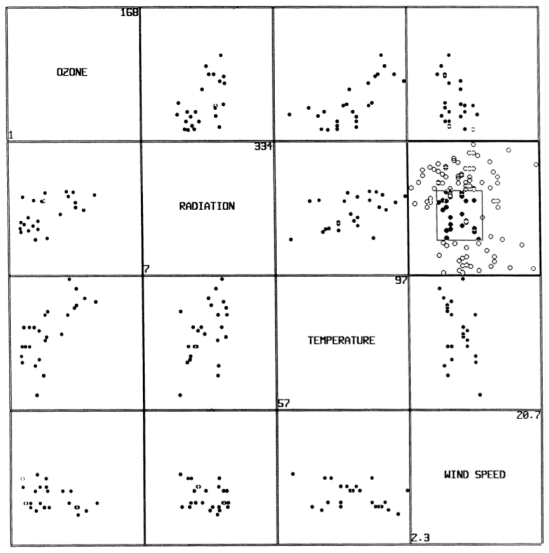
Figura 4.2: Comparación entre una visualización con un *scatterplot* tradicional y otro usando *sunflowers*. El conjunto de datos fue generado a partir de dos distribuciones binormales y una uniforme. (Página 38)



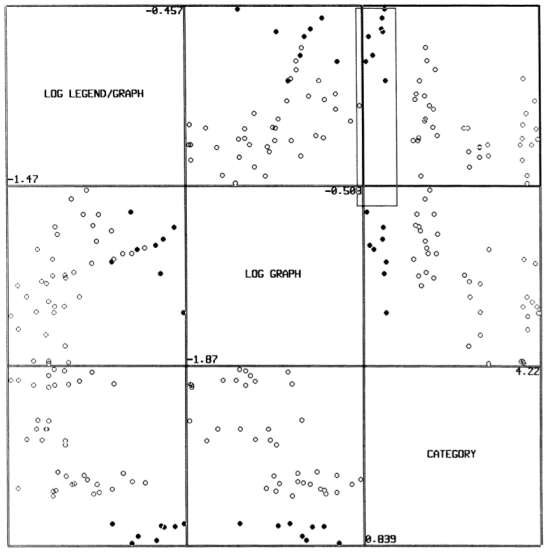
(a) Conjunto con 275 datos de 3 dimensiones. Todos los datos dentro del rectángulo se resaltan en todos los *scatterplots* de la matriz.



(b) Conjunto con 111 datos de 4 dimensiones. El resaltado persistente permite resaltar un grupo irregular de datos.

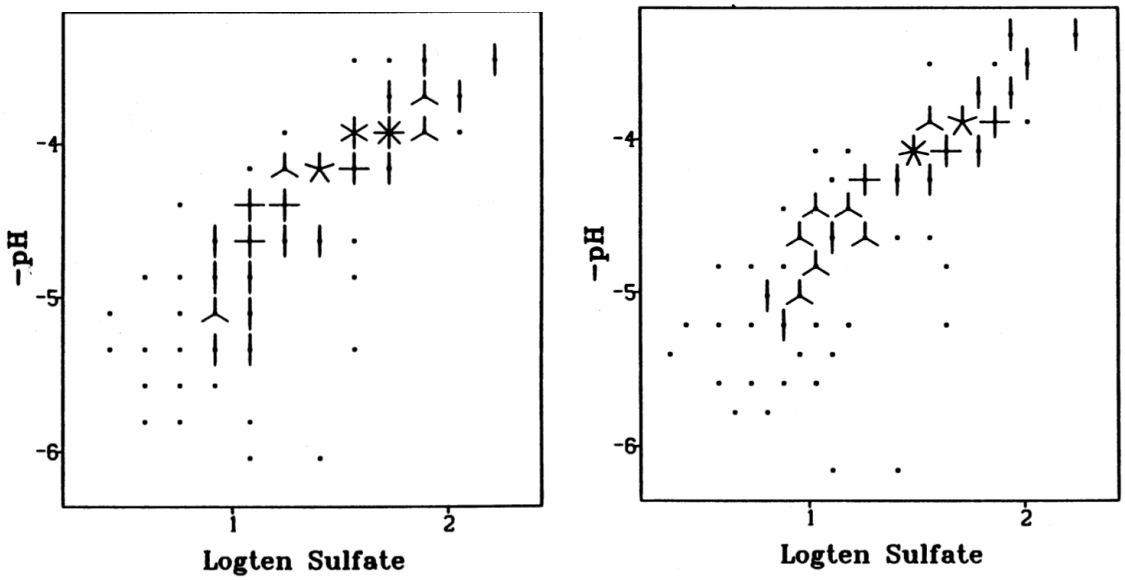


(c) Con el ocultamiento solamente los datos que se encuentran resaltados en el *scatterplot* activo son los que se muestran en los demás *scatterplots* de la matriz.



(d) El brushing definido también es aplicable en el caso de categorías.

Figura 4.3: Ejemplos de operaciones definidas con *brushing* sobre una matriz de *scatterplots*. (Página 38)



(a) *Sunflowers* [CM84]. El espacio se divide en una grilla cuadriculada. Cada celda se representa con una "flor" y la cantidad de pétalos de la flor es la cantidad de datos que hay en la celda. Cuando hay un solo dato se representa con un punto.

(b) La grilla es hexagonal: cada celda es un hexágono. Se pierden las direcciones horizontal y vertical y las celdas parecen estar más juntas.

Figura 4.4: Comparación del mismo conjunto de datos visualizado con *sunflowers* usando una grilla cuadriculada y una grilla hexagonal. (Página 39)

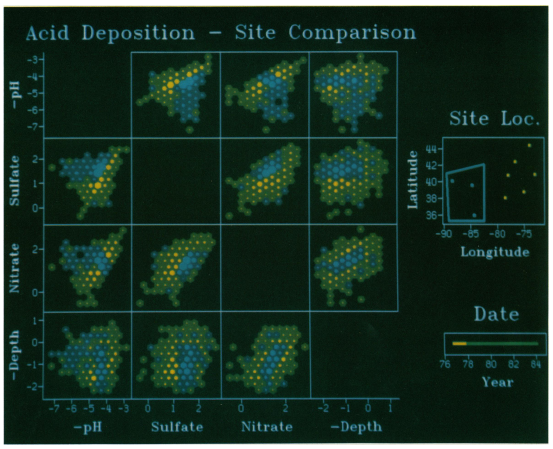
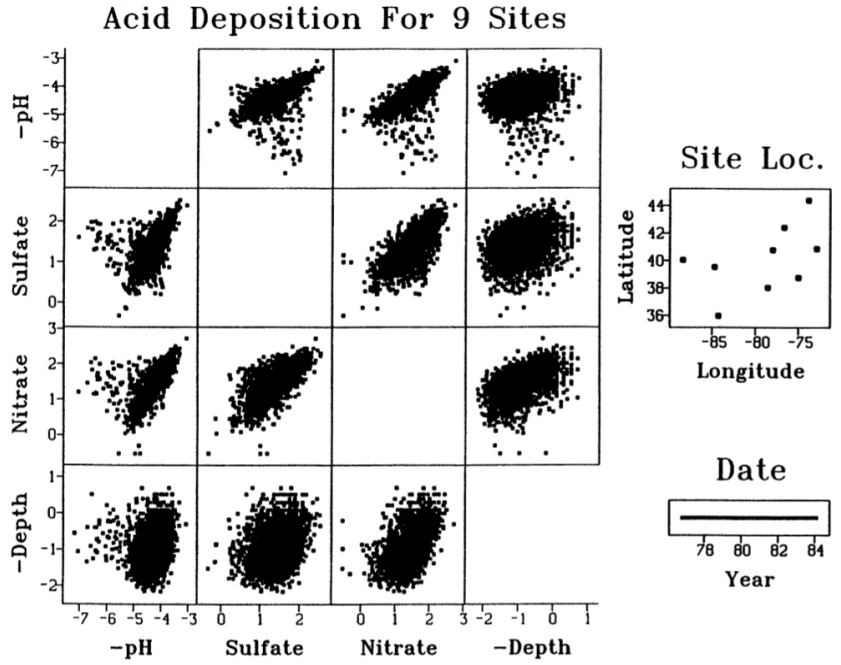
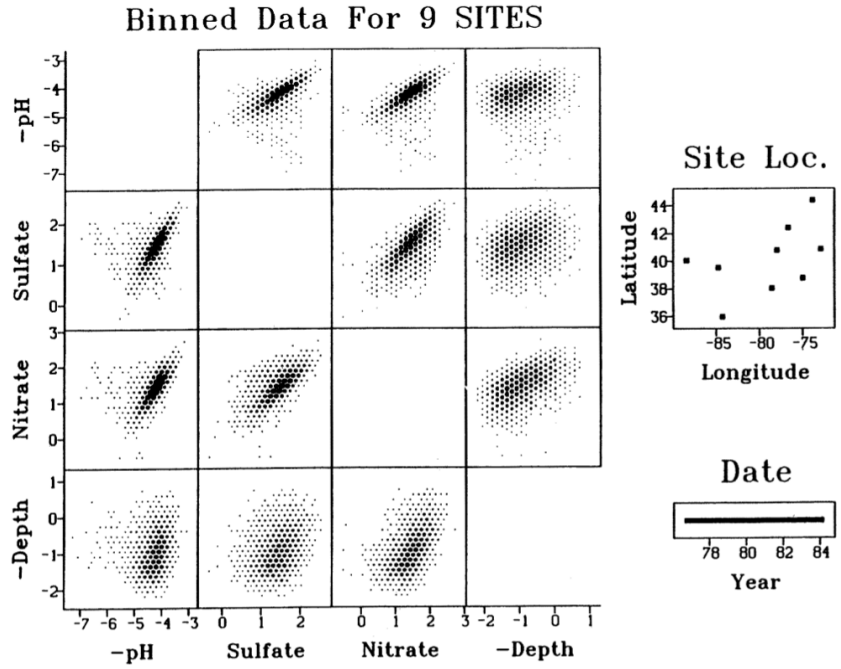


Figura 4.5: Selección y comparación de subconjuntos agregando color. Sobre el gráfico de ubicación de las ciudades se selecciona el subconjunto. Las regiones cian se corresponden con la más alta densidad de las ciudades seleccionadas, mientras que las regiones amarillas se corresponden con la más alta densidad de las ciudades no seleccionadas. (Página 39)



(a) Visualización tradicional de una matriz de *scatterplots*. La superposición es altamente notoria.



(b) Representación de la densidad a través de áreas hexagonales. El área de cada hexágono es proporcional a la cantidad de datos que hay en la celda. El hexágono más grande es del tamaño de la celda. Debajo de cierta cantidad, cualquier número se representa con el mismo hexágono pequeño. Las regiones de más alta densidad son fácilmente distinguibles.

Figura 4.6: Los datos visualizados son múltiples medidas de 41 000 muestras de lluvia en nueve ciudades en la red *Acid Deposition System*. A la derecha de cada matriz de *scatterplots*, se muestran dos gráficos adicionales, uno muestra la ubicación de la ciudades y el otro los años de las colecciones de datos. (Página 39)

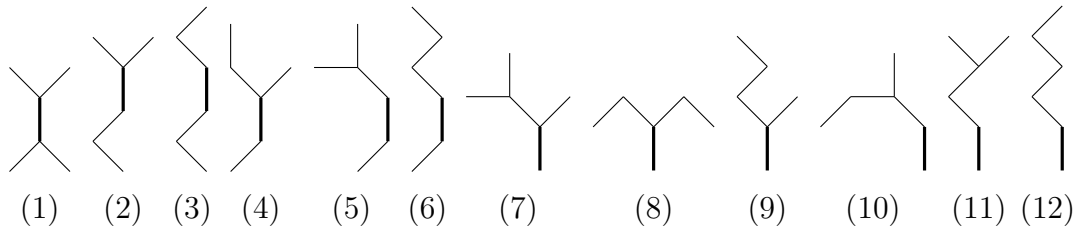


Figura 4.7: Familia de íconos que consiste de 12 íconos formados por 5 segmentos. (Página 39)

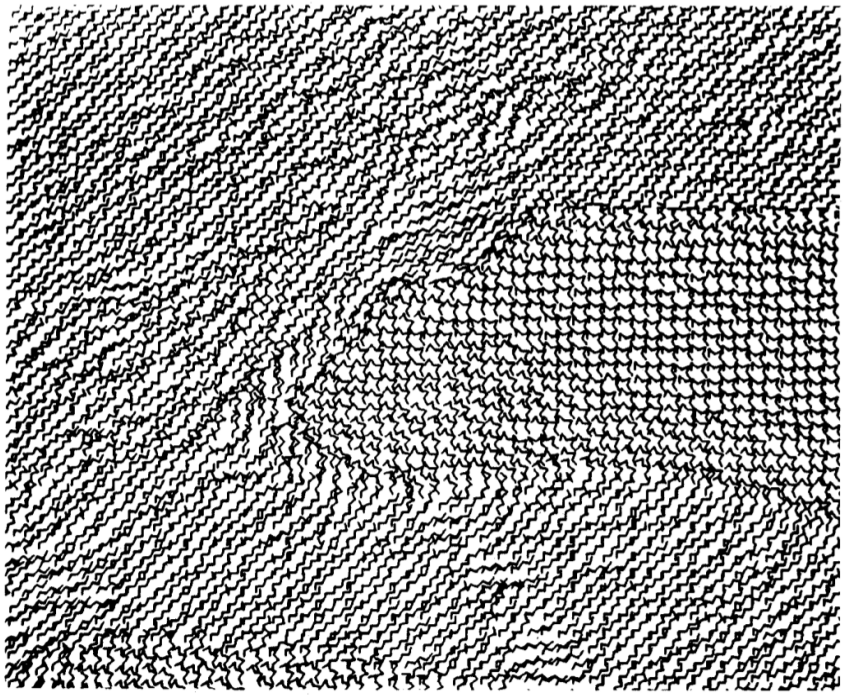


Figura 4.8: Imagen satelital multiespectro visualizada utilizando la familia de íconos de la figura 4.7. (Página 39)

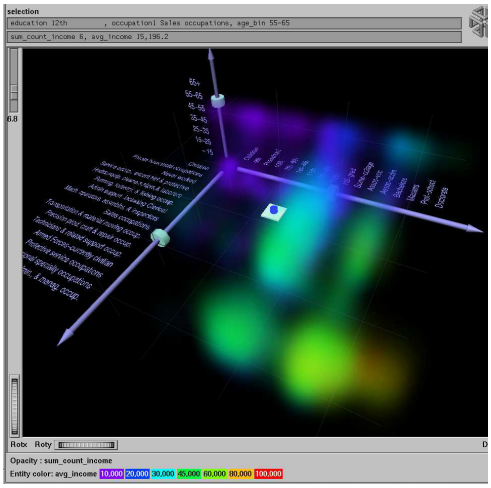
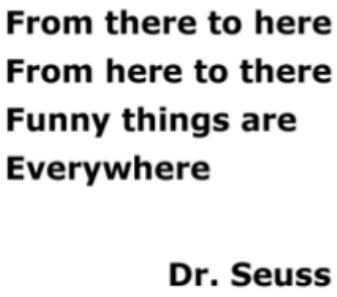


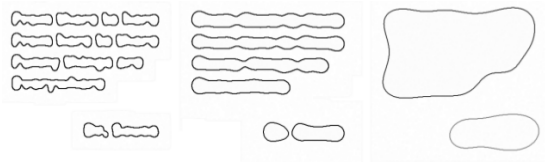
Figura 4.9: El factor de opacidad se redujo para mostrar solo las regiones más densas. La gráfica representa 149 643 datos. (Página 40)



(a) Imagen original

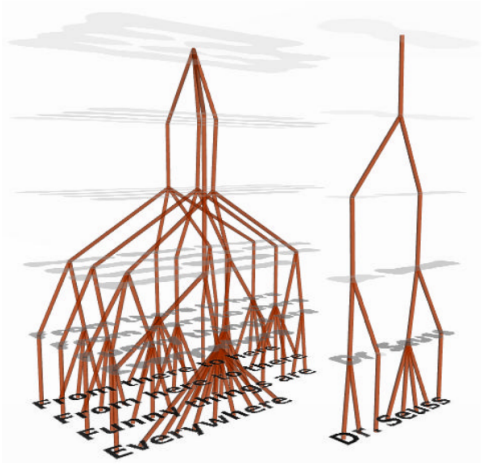
(b) Diferentes escalas

(c) Detección de bordes



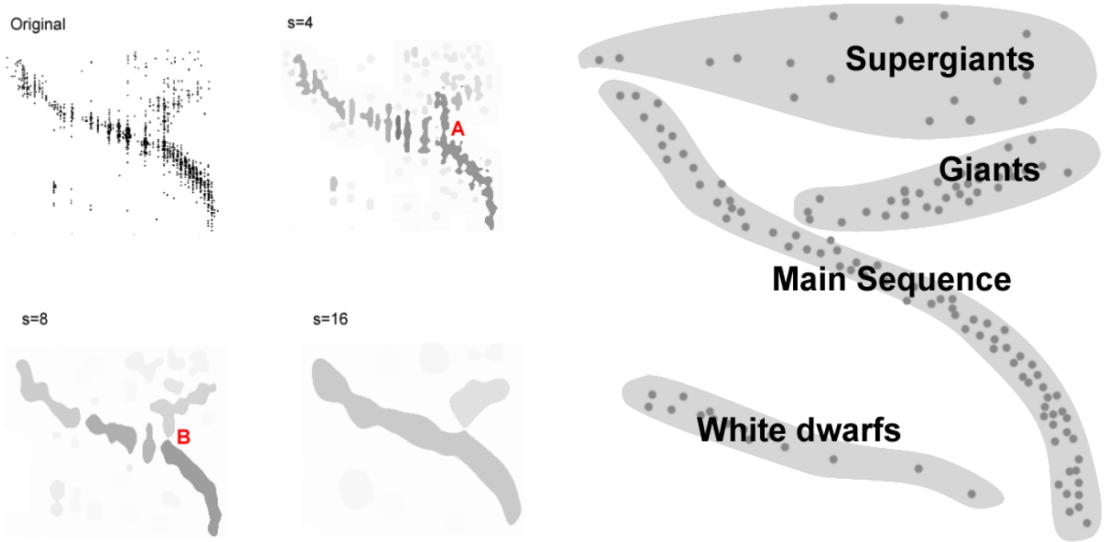
(d) Contornos

(e) Caricaturas Gestalt



(f) Enlaces entre las diferentes escalas

Figura 4.10: Modelo multi-escala. La figura (a) presenta la imagen original. En la figura (b) se muestran tres escalas diferentes de la misma imagen. En las figuras (c) y (d) se detectaron bordes en diferentes escalas y se delimitaron los contornos cerrados. En la figura (e) se muestran las caricaturas Gestalt finales. Finalmente en la figura (f) se ha armado la jerarquía entre las diferentes escalas. (Página 40)



(a) Caricaturas Gestalt del diagrama astronómico de Hertzprung-Russell.

(b) Interpretación del diagrama hecha por un astrónomo.

Figura 4.11: El diagrama de Hertzprung-Russell visualiza temperatura (eje x) y magnitud absoluta (eje y) de 300 estrellas. La figura (a) muestra las caricaturas Gestalt en diferentes escalas y la figura (b) indica cómo interpreta un experto el diagrama. (Página 40)

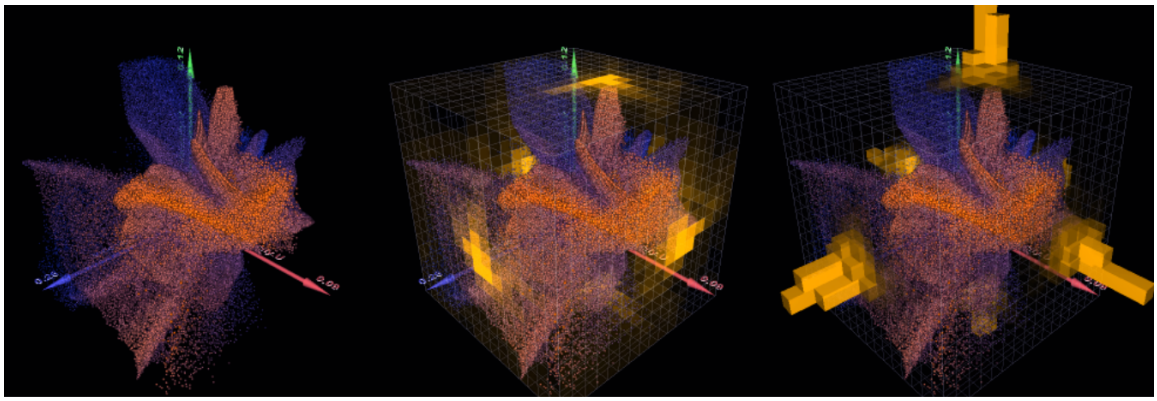


Figura 4.12: Representación de la densidad de puntos con histogramas 2D semitransparentes. En la figura de la izquierda, la densidad no es apreciable. En la figura del medio, se representan las densidades con texturas a través de la opacidad de la celda. En la figura de la derecha, la densidad en cada celda se mapea tanto a la altura de la barra como a su opacidad. (Página 41)

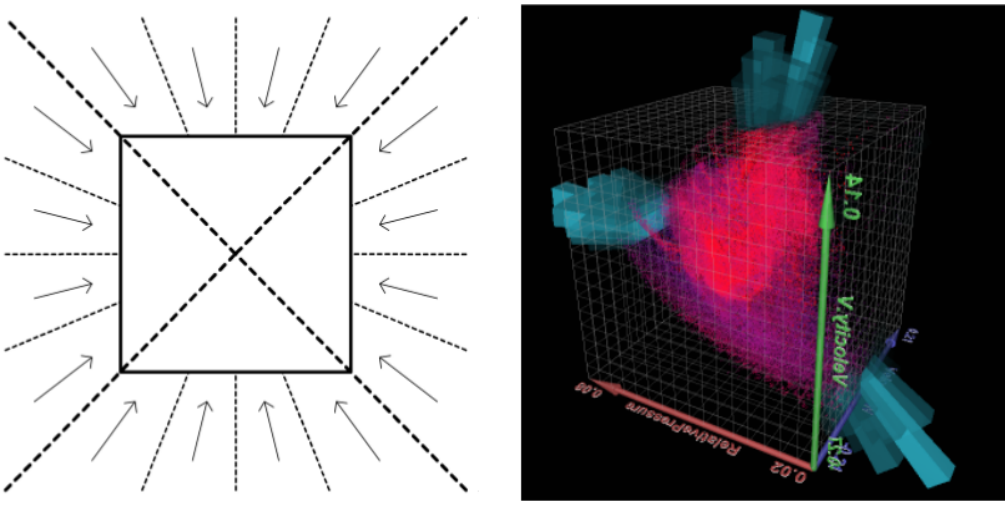


Figura 4.13: Representación del contexto. Las barras transparentes con proyección perspectiva desde el centro capturan todo el contexto espacial. (Página 41)

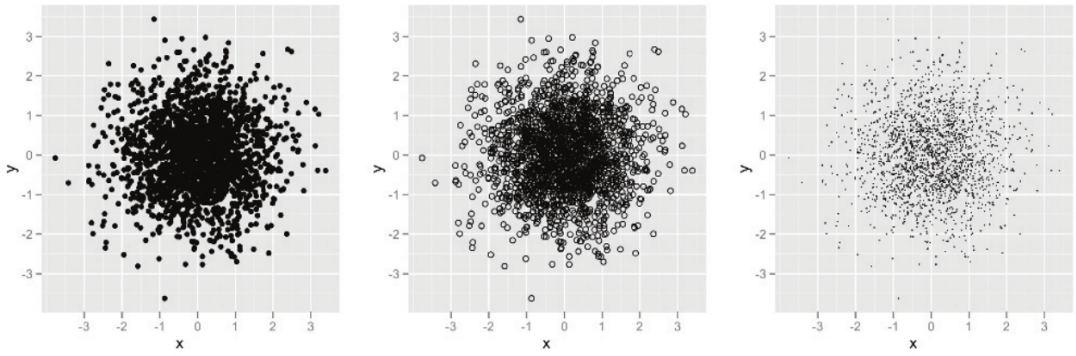


Figura 4.14: Solución para una superposición leve. De izquierda a derecha: *scatterplot* original, *scatterplot* con glifos huecos y *scatterplot* con un punto (un píxel) como glifo. (Página 41)

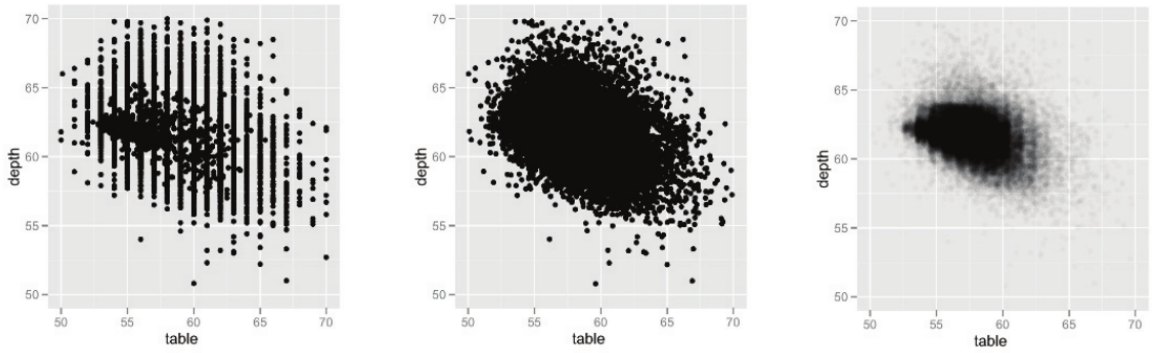


Figura 4.15: Se muestra el *jittering* y transparencias para aliviar la superposición en datos discretos. De izquierda a derecha: geometría original de los puntos, geometría con un *jittering* horizontal de 0,5 y geometría con un *jittering* horizontal de 0,5 y transparencia con un *alfa* de 1/50. (Página 42)

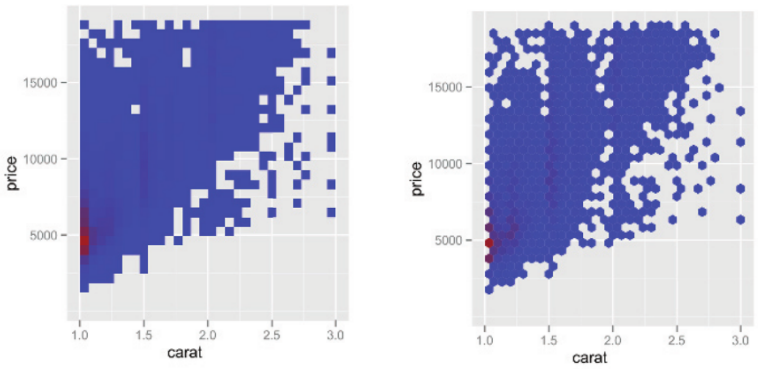


Figura 4.16: Celdas cuadradas vs. celdas hexagonales en R. La figura de la izquierda muestra un *scatterplot* usando celdas cuadradas, mientras que la figura de la derecha muestra los mismos datos usando celdas hexagonales. (Página 42)

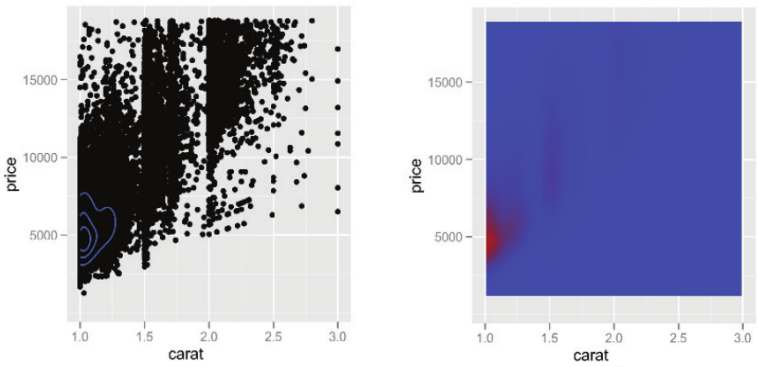


Figura 4.17: Estimación de la densidad para visualizar la densidad de puntos. La figura de la izquierda representa un *scatterplot* que permite visualizar la densidad a través de contornos. La figura de la derecha es un *scatterplot* que permite visualizar la densidad de los datos a través del color de los píxeles. (Página 42)

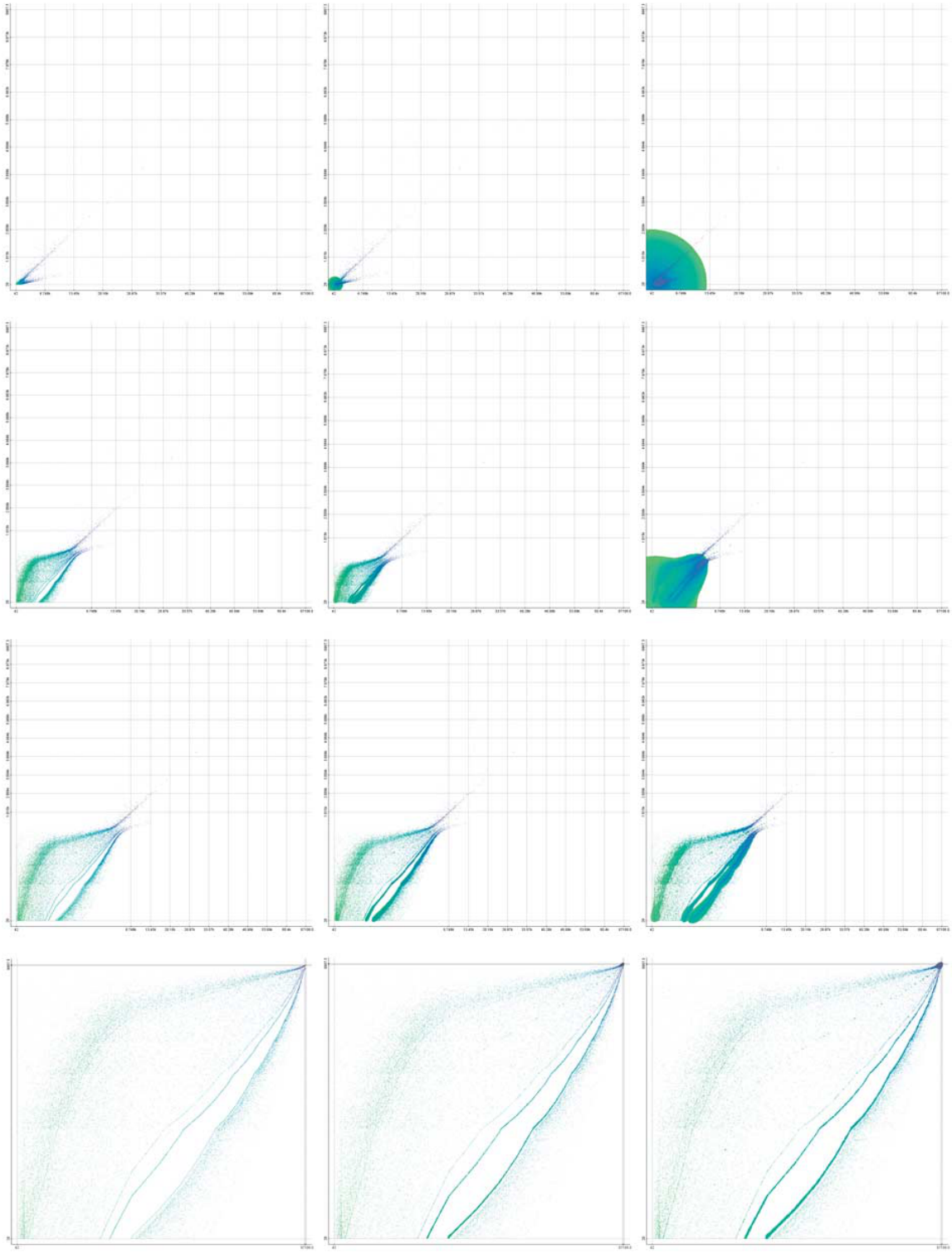


Figura 4.18: *Scatterplot* sin superposición de datos correspondientes a un Servicio Telefónico. El eje x representa la duración de la llamada, el eje y el costo del servicio y el color es el número de participantes. La representación original está en la esquina superior izquierda, la superposición disminuye de izquierda a derecha y la distorsión aumenta de arriba hacia abajo. (Página 42)

Capítulo 5

Predicción de la visibilidad en *scatterplots*

Se presenta una métrica que evalúa la visibilidad de los datos teniendo en cuenta la visibilidad de los glifos en el scatterplot*. Se definió una métrica que dados un conjunto de datos, el tamaño de la ventana y el tamaño del glifo que representará los datos, estima el porcentaje de glifos que serán siempre visibles independientemente del orden en que estos sean mostrados. Para definir y aproximar dicha métrica se experimentó con conjuntos de datos aleatorios seleccionados siguiendo distribuciones normales para ambas dimensiones de los datos.*

5.1. Introducción

Este capítulo se centrará en los *scatterplots* 2D, ya que son una técnica muy útil y ampliamente adoptada para la visualización de datos bidimensionales, extensible para datos multidimensionales y muy adecuada para la visualización de grandes volúmenes de datos.

El resultado de una visualización con *scatterplots* no depende únicamente del conjunto de datos, sino también de características particulares elegidas para la visualización: ¿cuál es el tamaño de la visualización? ¿de qué tamaño son los glifos? ¿qué forma tienen?

Ya se ha puntualizado la importancia de contar con métricas adecuadas que permitan determinar cuán escalables visualmente son las distintas técnicas. Por otro lado, el *scatterplot* es una técnica de visualización sumamente versátil y utilizada y, dado que no existen actualmente métricas orientadas a cuantificar la escalabilidad visual de los mismos, en este capítulo:

- Se propone una métrica que estima los glifos siempre visibles en un *scatterplot* independientemente del orden en que estos sean dibujados.
- Se presenta una aproximación matemática de la métrica en función de la cantidad de datos a visualizar, el tamaño de la ventana y el tamaño del glifo.

En la sección 5.4 se analizará cómo, a partir de un contexto determinado (datos, técnica y máximo espacio disponible), la medida definida puede asistir al usuario en la selección de los parámetros para que la visualización resulte en la *mejor vista posible* en cuanto a la visibilidad de los nodos se refiere.

5.2. Antecedentes

Dada la necesidad de las métricas, varios autores han trabajado en la definición de métricas sobre diferentes visualizaciones y en particular sobre los *scatterplots*. Si bien dichas métricas no están enfocadas particularmente en la escalabilidad visual, miden aspectos (como la oclusión o la degradación de la visualización) que impactan directamente en la escalabilidad visual de una técnica.

Brath [Bra97] propone conceptualmente medidas para ayudar en el diseño y en la evaluación de visualizaciones 3D estáticas, aplicables a *scatterplots*:

- *Número de puntos de datos.* Cantidad de valores discretos representados en la pantalla en un instante.
- *Densidad de datos* Razón entre la cantidad de datos y el número de píxeles en la ventana, donde la ventana no incluye barras de herramientas, menús, bordes, etc.
- Medidas de complejidad cognitiva.
  - Número de dimensiones simultáneas.
  - Máximo número de dimensiones por cada representación de tareas separables.
  - *Efectividad.* Para cuantificar la efectividad de la representación crearon un modelo simple de la complejidad cognitiva a través de un esquema de puntos. Este modelo califica la efectividad del mapeo de dimensiones de los datos a atributos visuales, siendo mayor el puntaje cuánto más esfuerzo cognitivo demanda el mapeo:

Mapeo	Puntaje
n–a–1	$3 \times n$
1–a–1 general	2
1–a–1 intuitivo	1
representación preexistente	0

- *Porcentaje de oclusión.*

$$\frac{\text{número de puntos de datos completamente tapados}}{\text{número total de puntos de datos}}$$

- *Porcentaje de puntos identificables.*

$$\frac{\begin{array}{c}\text{número de puntos de datos visibles e identificables}\\ \text{en relación con todos los puntos de datos visibles}\end{array}}{(\text{número de puntos de datos visibles})^2}$$

Basándose en las medidas de contenidos de información desarrolladas por Shannon y en la teoría de comunicaciones, Yang-Pelaez y Flowers [YPF00] desarrollaron medidas para evaluar la efectividad de visualizaciones; estas medidas se relacionan con *contenido de información en los datos*, *contenido de información en la visualización*, *capacidad de información de una visualización* y *contenido de información topológica*. Cada dimensión contribuye al total de la información tanto de los datos como en la visualización en $\log_2\left(\frac{\text{rango}}{\text{precisión}}\right)$ y establecen que una visualización es efectiva si el contenido de información de los datos y de la visualización son iguales.

Bertini y Santucci [BS04; BS06a] se concentraron en determinar el correcto muestreo de los datos para garantizar automáticamente parámetros de calidad en una visualización con *scatterplots* 2D y glifos de un único píxel. Para esto estiman la cantidad de píxeles activos[1], el espacio libre disponible y las colisiones como resultado de una dada cantidad de muestras del conjunto real de datos. Esta predicción la utilizan para realizar el muestreo de los datos. Para estimar los píxeles libres y las colisiones, calculan la probabilidad de tener exactamente k colisiones graficando n puntos en un área pequeña de p píxeles

$$\frac{\text{cantidad de configuraciones con exactamente } k \text{ colisiones}}{\text{cantidad total de configuraciones}}.$$

A partir de esta probabilidad se puede calcular:

- La media de los elementos que colisionan (como un porcentaje de n).
- El espacio ocupado (como un porcentaje de p).
- El espacio libre (como un porcentaje de p).

La visualización final se divide en pequeñas áreas de p píxeles y se calculan varias medidas para evaluar diferentes aspectos de la calidad de la imagen:

1. Degradación:
   - *Razón entre los puntos de datos y los píxeles activos.* Da una medida de cuánta superposición hay en la pantalla.
   - *Razón entre la cantidad de colisiones y los puntos de datos.* Es el número de colisiones k por área de muestreado dividido por la cantidad de datos a mostrar.

[1] Los píxeles activos son aquellos que se distinguen del color de fondo.

- *Porcentaje de la ventana con superposición no aceptable.* Provee una medida global sobre la degradación de la imagen.
- *Razón entre los puntos de datos en áreas superpobladas y los puntos de datos.* Da una medida del porcentaje de datos pertenecientes a las áreas superpobladas.

2. Diferencias de densidades:

   - *Razón entre la densidad de datos y la densidad de píxeles activos.* Da una medida de la preservación de densidades relativas entre las diferentes zonas de la imagen.

3. Efectos negativos luego del muestreo de los datos:

   - *Razón de áreas borradas.* En función de las pequeñas áreas de p píxeles, da una medida de la porción de la pantalla que fue vaciada como efecto del muestreo de datos:

$$\frac{\begin{array}{c}\text{número áreas vacías luego del muestreo de datos} \\ -\text{número áreas vacías antes del muestreo de datos}\end{array}}{\text{cantidad de áreas}}$$

Por otro lado, en el ámbito del Análisis Visual se han desarrollado las *scagnostics* (*scatterplots diagnostics*) como medidas para interpretar la información ([Wil+05]) ya que ofrecen la posibilidad de detectar anomalías en grandes matrices de *scatterplots*. Estas medidas se derivan de varias características de los grafos:

- La longitud de un arco es la distancia euclídea entre los vértices.
- La longitud de un grafo es la suma de la distancias de todos sus arcos.
- Un camino es una lista de vértices tal que todo par de vértices adyacentes en la lista es un arco.
- Un camino es cerrado si el primero y el último vértice son el mismo.
- Un camino cerrado es el contorno de un polígono.
- El perímetro de un polígono es la longitud de su contorno.
- El área de un polígono es el área de su interior.
- El diámetro de un grafo es el más largo de los caminos más cortos en el grafo.

A partir del *convex hull*, *nonconvex hull (alpha hull)* y el *minimal spanning tree* y las características enumeradas, calculan índices para detectar anomalías en los datos a través de la visualización de grandes matrices de *scatterplots*:

- *Valores atípicos.* Mide la razón entre el total de las longitudes de los arcos del *minimal spanning tree* y las longitudes de los arcos extremadamente largos conectados a puntos de grado uno.
- *Forma.* Buscan medir la forma del conjunto de datos, tanto los aspectos topológicos como geométricos.
  - *Convexidad.* Es la razón entre el área del *alpha hull* y el *convex hull*.
  - *Delgadez.* Es la razón corregida y normalizada entre el perímetro y el área del *alpha hull*.
  - *Fibroso.* La forma fibrosa es una forma delgada y sin ramificaciones. Es la razón entre el diámetro y la longitud del *minimal spanning tree*.
  - *Rectitud.* Es la razón entre la distancia euclídea entre los extremos del camino que define el diámetro del *minimal spanning tree* y el diámetro del *minimal spanning tree*.
- *Densidad.* Detectan diferentes distribuciones de los puntos.
  - *Sesgo.* La distribución de las longitudes de los arcos del *minimal spanning tree* da información sobre la densidad relativa de los puntos en el *scatterplots*.
  - *Grumos.* Para detectar *clusters* de puntos se basan en el isomorfismo entre los *dendrogramas*[2] y el *minimal spanning tree*.
  - *Coherencia.* Se define coherencia en un conjunto de puntos como la presencia de caminos relativamente suaves en el *minimal spanning tree*. Esto se corresponde con funciones algebraicas suaves, series temporales y curvas, incluso puntos que formen campos vectoriales.
  - *Estrías.* Reconoce configuraciones de puntos que sigan caminos lineales, representen campos vectoriales o texturas estriadas.
- *Tendencia* Si el conjunto de puntos es una función de x, utilizan el cuadrado del coeficiente de correlación de Spearman para evaluar la *monoticidad* de los puntos.

Dang y Wilkinson [DW14] trabajaron con estas medidas en grandes matrices de *scatterplots* con más de 100 dimensiones, lo que les permitió localizar anomalías o identificar distribuciones similares entre los diferentes *scatterplots*.

5.3. Una métrica para cuantificar la visibilidad

Hasta ahora, las medidas definidas sobre los *scatterplots* se concentran en medir características de visualizaciones ya *renderizadas*. Por otro lado, la *razón entre la cantidad de*

[2] Un *dendrograma* es un diagrama de árboles utilizado para ilustrar el orden de los *clusters* producidos a través de *clustering* jerárquico.

colisiones y los puntos de datos ([BS04; BS06a]) considera glifos de un único píxel, dejando de lado el análisis de uno de los aspectos más importantes que afecta la superposición en los *scatterplots*. Las *scagnostics* se enfocan en la extracción y deducción de información de un conjunto de datos a partir de una visualización y no en cuantificar aspectos de la visualización en sí misma.

Nuestro objetivo es definir una medida que *prediga*, sin necesidad de *renderizar* la visualización, cuán *buena* será la visualización resultante. Dado que la superposición es un importante factor limitante de los *scatterplots*, se buscó una medida que exprese matemáticamente el concepto de *visibilidad*, es decir lo opuesto al concepto de *porcentaje de oclusión* definido en [Bra97] (y descripto en la sección 5.2)

$$visibilidad = 1 - porcentaje\ de\ oclusión$$

y además tenga en cuenta los parámetros de la visualización

$$visibilidad = f(parámetros\ visualización, datos).$$

En esta sección se define y formaliza el concepto del *índice de visibilidad* presentando un método para calcularlo. Para ello, y de acuerdo con las guías de Tatu et al. [Tat+10] presentadas en la sección 3.3 se desarrolló un algoritmo para calcular el índice de visibilidad de visualizaciones con *scatterplots*. Luego, se define un modelo matemático que aproxima dicho índice y permite predecir el resultado de la visualización.

Para definir la medida se consideraron los *scatterplot* 2D, es decir un espacio de dos dimensiones en el cual se dibujan glifos; cada glifo representa un ítem de datos cuya posición se calcula mapeando dos atributos de los datos en las coordenadas espaciales.

5.3.1. Índice de visibilidad

El *índice de visibilidad* se define como una medida específica para los *scatterplots* que, dado un conjunto de datos y las dimensiones de la ventana y del glifo, estima el porcentaje esperable de glifos que no se encuentran completamente superpuestos con otros glifos (existe al menos un píxel no superpuesto con otro glifo), es decir la cantidad esperable de glifos que serán siempre visibles independientemente del orden en que sean dibujados.

Esta medida es un criterio de *organización espacial* (Freitas et al. [Fre+02]) y mide la calidad de la imagen en el espacio de datos (características de los datos: distribución, cantidad) y de la imagen (características de la visualización: tamaño, glifo) con el propósito de conseguir un mapeo visual adecuado (Bertini et al. [Ber+11]).

Adoptando la convención de Brath [Bra97], las dimensiones de la ventana (alto y ancho) incluyen exclusivamente al área donde se grafica el *scatterplot*, es decir no incluyen menúes, bordes, botones, visualizaciones accesorias, etc. Dado que para el análisis se consideran únicamente ventanas cuadradas, con un solo valor se puede representar el alto y el ancho de la ventana y por lo tanto su tamaño. Los glifos también se consideran

cuadrados y, por lo tanto, también alcanza con un único valor para representar su tamaño. En ambos casos se considera el lado del cuadrado como el tamaño del glifo o de la ventana.

5.3.2. Cálculo del índice de visibilidad. Algoritmo

El índice de visibilidad τ de un conjunto con cantidad de n datos visualizado con un *scatterplot* se puede calcular mediante el siguiente algoritmo:

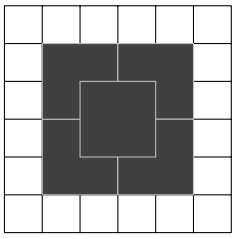
1. Crear una matriz de enteros M que representará la visualización resultante, un entero por cada píxel, inicializada en ceros.
2. Crear una lista de glifos G. Cada glifo deberá tener información que permita obtener las posiciones (píxeles) exactas que ocupa en la matriz de enteros M.
3. Por cada dato del conjunto:
   a) Crear un glifo g.
   b) Agregar g a la lista G.
   c) Ubicar g en la matriz e incrementar en 1 cada posición que ocupe en M.
4. La cantidad de glifos siempre visibles es la cantidad de glifos g en G tales que alguno de los píxeles que le correspondan en la visualización esté asociado a un único glifo, es decir, alguna de la posiciones que ocupe en la matriz M tenga valor 1.
5. El índice de visibilidad es la razón entre la cantidad de glifos siempre visibles y la cantidad total de glifos.

En la figura 5.1 se muestran dos ejemplos de glifos ubicados en el *scatterplot* y la matriz de enteros correspondiente a cada uno.

5.3.3. Modelo matemático del índice de visibilidad

Para analizar el comportamiento de la métrica definida se experimentó con 120 conjuntos de datos generados aleatoriamente siguiendo 16 distribuciones normales diferentes para cada una de las dos dimensiones y con 23 cantidades diferentes de datos ($10; 10^{1,25}; 10^{1,5}; 10^{1,75}; 10^2 \ldots 10^6; 10^{6,25}$ y $10^{6,5}$). Cada uno de estos 120 conjuntos de datos se visualizó con 375 *scatterplots* de diferentes configuraciones: 25 tamaños diferentes de ventana ($100; 300; 500 \ldots 4700$ y 4900 píxeles de lado) y 15 tamaños distintos de glifos ($2; 4; 6 \ldots 28$ y 30 píxeles de lado). Las 16 distribuciones distintas se generaron con media 1 y diferentes desvíos estándares ($0{,}05; 0{,}08; 0{,}1; 0{,}3; 0{,}5; 0{,}8; 1; 3; 5; 8; 10; 30; 50; 80; 100$ y 300).

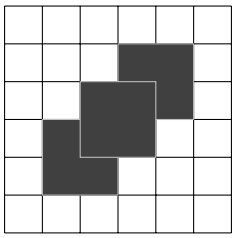
Se utilizó la distribución normal para llevar a cabo los experimentos dado que esta distribución es apropiada para la mayoría de los fenómenos naturales, cuando la muestra es muy grande y el error es lo suficientemente pequeño.



0	0	0	0	0	0
0	1	1	1	1	0
0	1	2	2	1	0
0	1	2	2	1	0
0	1	1	1	1	0
0	0	0	0	0	0

cantidad total de glifos	5
cantidad de glifos siempre visibles	4
índice de visibilidad	$4/5 = 0{,}8$

(a)



0	0	0	0	0	0
0	0	0	1	1	0
0	0	1	2	1	0
0	1	2	1	0	0
0	1	1	0	0	0
0	0	0	0	0	0

cantidad total de glifos	3
cantidad de glifos siempre visibles	3
índice de visibilidad	$3/3 = 1$

(b)

Figura 5.1: Ejemplos de glifos ubicados en el *scatterplot* con su matriz de enteros correspondiente. En la figura 5.1a se muestra un caso en el cual hay un glifo que será visible dependiendo del orden en que sea dibujado: si se dibuja primero el glifo central y luego los demás glifos, el primero quedará oculto detrás de los demás; sin embargo, si el glifo central se dibuja último es posible que sea visible, dependiendo de los colores del glifo (contorno diferente, colores diferentes, etc.). En la figura 5.1b se muestra un caso en el cual, independientemente del orden en que se dibujen los glifos, va a ser visible al menos un píxel de cada glifo.

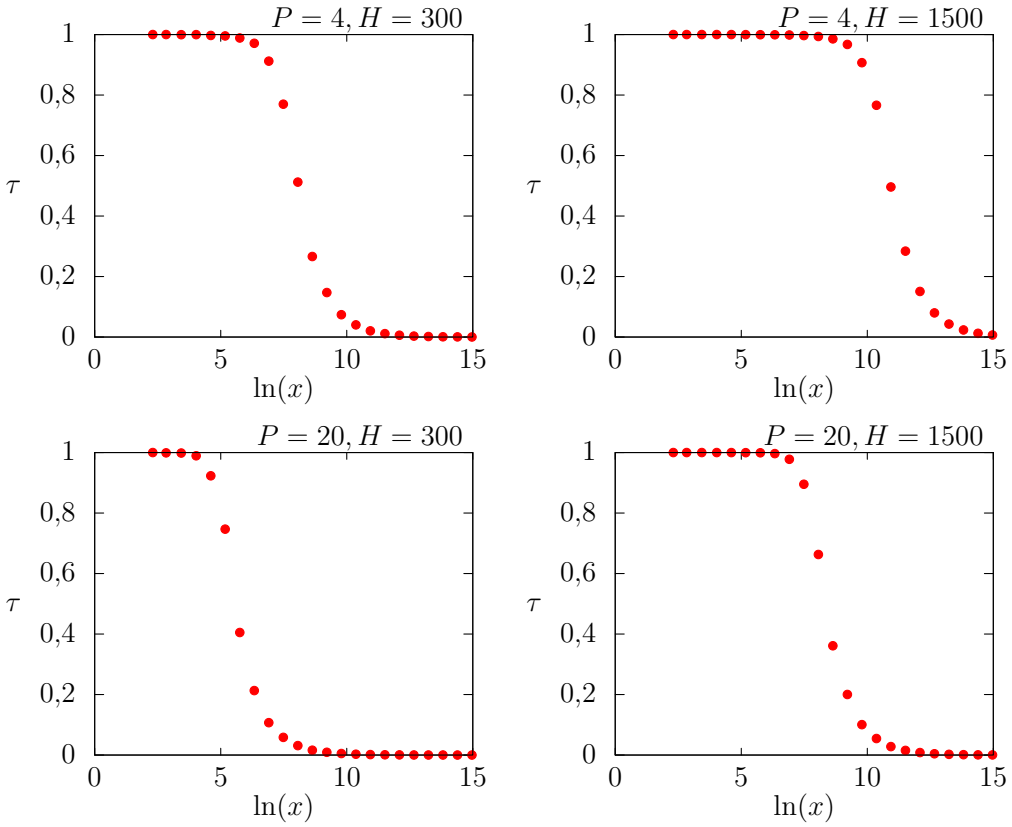


Figura 5.2: Comportamiento del índice de visibilidad en función del logaritmo de la cantidad de datos. P y H representan el tamaño del glifo y de la ventana, respectivamente y x es la cantidad de datos del conjunto.

Para cada terna (cantidad de datos, tamaño de la ventana, tamaño del glifo) se obtiene un rango de valores (diferentes valores para cada experimento con las 16 distribuciones diferentes), por lo tanto se analizó para cada par ⟨tamaño de ventana, tamaño de glifo⟩ cómo varía el promedio del índice de visibilidad en función de la cantidad de datos x (ver figura 5.2).

Características de la función f

Teniendo en cuenta que h, p y x son variables independientes entre sí, la función f que aproxime al índice de visibilidad τ debe cumplir las siguientes características:

1. $\lim_{x\to 0} f(x,h,p) = 1$. A medida que la cantidad de datos a visualizar tiende a cero, mayores serán las *posibilidades* de que todos los datos sean siempre visibles independientemente del tamaño de la ventana o del glifo.

2. $\lim_{x\to \infty} f(x,h,p) = 0$. A medida que la cantidad de datos tiende a infinito, las *posibilidades* de que algún ítem de datos sea siempre visible tiende a cero.

3. $\lim_{h\to 0} f(x,h,p) = 0$. A medida que el área destinada a la visualización tiende a cero la proporción de datos siempre visibles tiende a cero.

4. $\lim_{h\to\infty} f(x,h,p) = 1$. A medida que el área destinada a la visualización tiende a infinito, la proporción de datos siempre visibles tiende a uno, independientemente de la cantidad de datos o del tamaño del glifo.

5. $\lim_{p\to 0} f(x,h,p) = 1$. A medida que se reduce el tamaño del glifo que representa los datos, la proporción de datos siempre visibles tiende a uno.

6. $\lim_{p\to\infty} f(x,h,p) = 0$. A medida que crece el tamaño del glifo, la proporción de datos siempre visibles tiende a cero, dado que las *posibilidades* de que algún ítem de datos sea siempre visible tiende a cero.

Definición de la función f

Para cada terna $\langle x,h,p\rangle$ se aproxima el índice de visibilidad τ con la función f

$$\tau \approx f(x,h,p) = \frac{1}{1+e^{\gamma(x,h,p)}}.$$

Luego, para que la función f satisfaga las condiciones enumeradas en la sección 5.3.3, la función γ debe satisfacer:

1. Si $\lim_{x\to 0} f(x,h,p) = \lim_{x\to 0} \frac{1}{1+e^{\gamma(x,h,p)}} = 1$, entonces $\lim_{x\to 0} \gamma(x,h,p) = -\infty$.
2. Si $\lim_{x\to\infty} f(x,h,p) = \lim_{x\to\infty} \frac{1}{1+e^{\gamma(x,h,p)}} = 0$, entonces $\lim_{x\to\infty} \gamma(x,h,p) = +\infty$.
3. Si $\lim_{h\to 0} f(x,h,p) = \lim_{h\to 0} \frac{1}{1+e^{\gamma(x,h,p)}} = 0$, entonces $\lim_{h\to 0} \gamma(x,h,p) = +\infty$.
4. Si $\lim_{h\to\infty} f(x,h,p) = \lim_{h\to\infty} \frac{1}{1+e^{\gamma(x,h,p)}} = 1$, entonces $\lim_{h\to\infty} \gamma(x,h,p) = -\infty$.
5. Si $\lim_{p\to 0} f(x,h,p) = \lim_{p\to 0} \frac{1}{1+e^{\gamma(x,h,p)}} = 1$, entonces $\lim_{p\to 0} \gamma(x,h,p) = -\infty$.
6. Si $\lim_{p\to\infty} f(x,h,p) = \lim_{p\to\infty} \frac{1}{1+e^{\gamma(x,h,p)}} = 0$, entonces $\lim_{p\to\infty} \gamma(x,h,p) = +\infty$.

Luego, considerando $\gamma(x,h,p) = a\ln(x) + b\ln(h) + c\ln(p) + d$ se tiene que:

1. $\lim_{x\to 0} \gamma(x,h,p) = [\lim_{x\to 0} a\ln(x)] + b\ln(h) + c\ln(p) + d = -\text{signo}(a)\infty$
2. $\lim_{x\to\infty} \gamma(x,h,p) = [\lim_{x\to\infty} a\ln(x)] + b\ln(h) + c\ln(p) + d = \text{signo}(a)\infty$
3. $\lim_{h\to 0} \gamma(x,h,p) = a\ln(x) + [\lim_{h\to 0} b\ln(h)] + c\ln(p) + d = -\text{signo}(b)\infty$
4. $\lim_{h\to\infty} \gamma(x,h,p) = a\ln(x) + [\lim_{h\to\infty} b\ln(h)] + c\ln(p) + d = \text{signo}(b)\infty$
5. $\lim_{p\to 0} \gamma(x,h,p) = a\ln(x) + b\ln(h) + [\lim_{h\to 0} c\ln(p)] + d = -\text{signo}(c)\infty$
6. $\lim_{p\to\infty} \gamma(x,h,p) = a\ln(x) + b\ln(h) + [\lim_{h\to 0} c\ln(p)] + d = \text{signo}(c)\infty$

por lo tanto, los coeficientes a, b y c deben satisfacer que:

$$a > 0, \quad b < 0 \text{ y } c > 0.$$

Nota sobre f y γ Si bien para $\gamma(x,h,p) = a\ln(x) + b\ln(h) + c\ln(p) + d$ y $f(x,h,p) = \frac{1}{1+e^{\gamma(x,h,p)}}$ se tiene que:

$$f(x,h,p) = \frac{1}{1+e^{\gamma(x,h,p)}} = \frac{1}{1+e^{a\ln(x)+b\ln(h)+c\ln(p)+d}} = \\ = \frac{1}{1+x^a h^b p^c e^d}.$$

Numéricamente, se obtienen mejores resultados a partir de la primera expresión para f que de la última expresión equivalente obtenida.

Aproximación de los coeficientes a, b, c y d

El índice τ se aproxima por partes, comenzando por a, considerando una función $\gamma(x,h,p) = g_1(x) = a\ln(x) + b_0$. El nuevo coeficiente b_0 se aproxima con una función $g_2(h) = b\ln(h) + c_0$ y finalmente, el coeficiente intermedio c_0 se aproxima con una función $g_3(p) = c\ln(p) + d$.

Esquemáticamente, la función g se divide de la siguiente forma para llevar a cabo la aproximación:

$$\gamma(x,h,p) = g_1(x) = a\ln(x) + \underbrace{b_0}$$
$$g_2(h) = b\ln(h) + \underbrace{c_0}$$
$$g_3(p) = c\ln(p) + d$$

En la figura 5.3 se muestran los pasos seguidos para aproximar el índice τ con la función $1/1 + e^{\gamma(x,h,p)}$ y obtener los valores de los coeficientes a, b, c y d; además se muestran ejemplos de los valores obtenidos en cada uno de los pasos. En todos los casos se utilizó `gnuplot`[3] para realizar las aproximaciones, con valores iniciales en 1. `gnuplot` utiliza una implementación del algoritmo no lineal de mínimos cuadrados de Marquardt-Levenberg.

Análisis del error

Para analizar el error de la aproximación f se comparan tanto los errores absolutos como los errores máximos entre f y τ.

Error	Mínimo	Máximo	Promedio
Absoluto	0,0	0,165811	0,012194
Relativo	0,0	0,999979	0,282614

Error medio cuadrático = 0,003646

Hay que destacar que, a pesar de tener un elevado error relativo máximo, estos valores se corresponden a valores de τ muy cercanos a cero, mientras que los valores de error absoluto máximo se corresponden al transitorio entre 1 y 0 (ver figura 5.4).

[3] `http://www.gnuplot.info/`

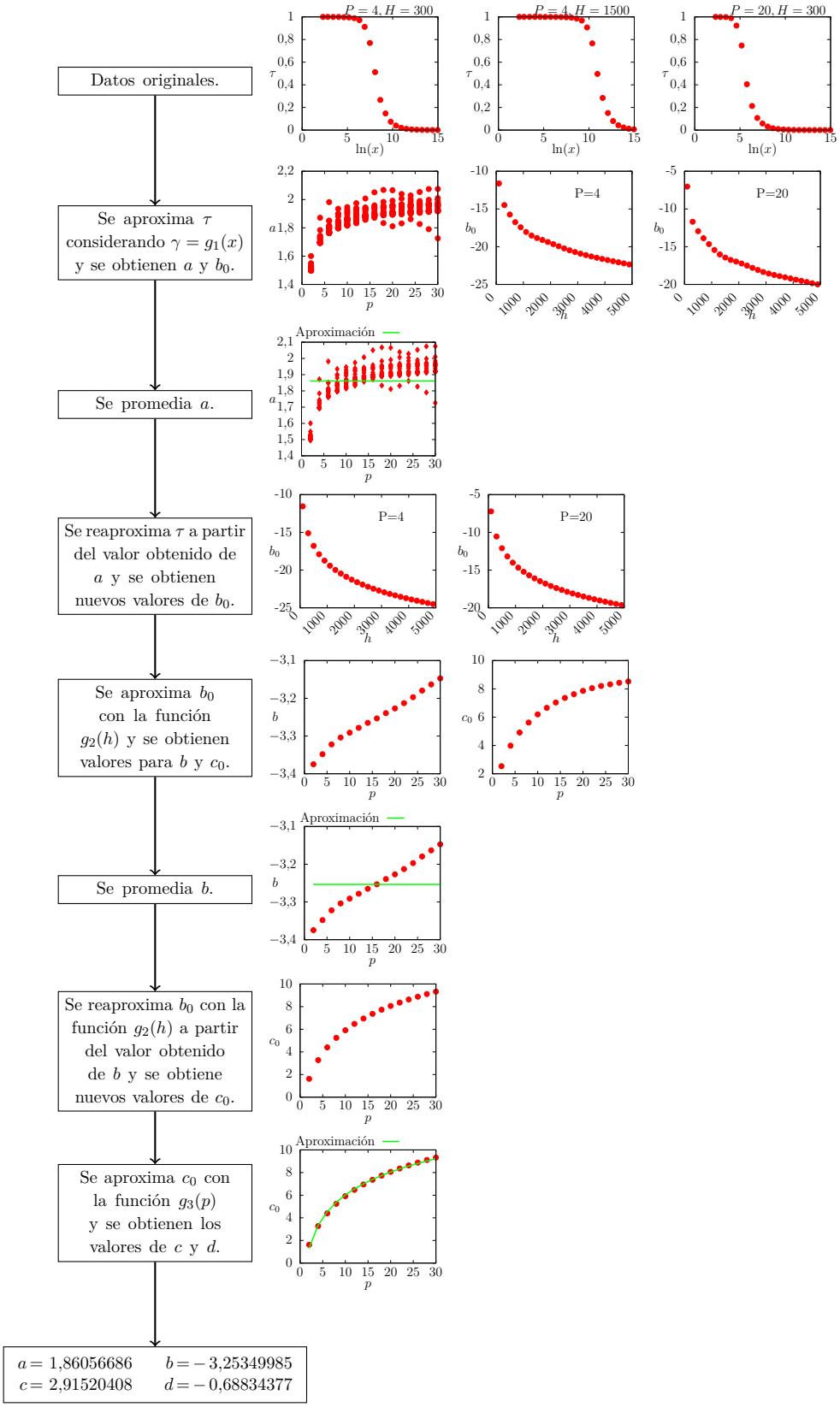


Figura 5.3: Pasos seguidos para obtener los coeficientes a, b, c y d de la aproximación de τ.

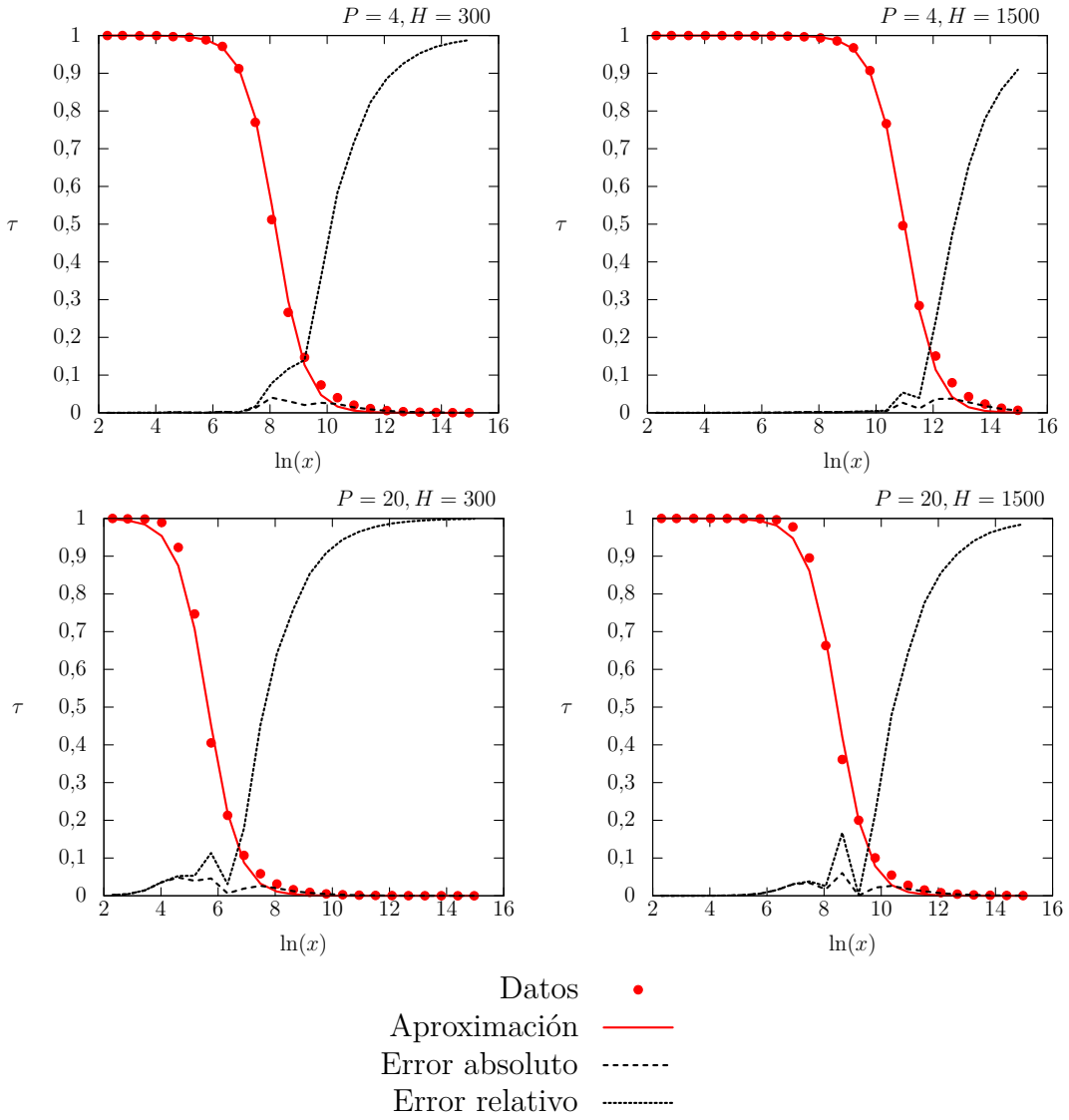


Figura 5.4: Gráfica de los errores relativos y absolutos entre f y τ.

5.4. Cómo la métrica asiste al usuario en la configuración de un *scatterplot*

Esta métrica tiene como objetivo, teniendo en cuenta la cantidad de datos a visualizar, guiar al usuario en la elección de los parámetros de la visualización, en particular en la elección del tamaño de la visualización y el tamaño del glifo. Si bien las fórmulas contemplan visualizaciones tan grandes o glifos tan pequeños como sea necesario, en la práctica el tamaño de la visualización no podrá ser mayor al tamaño de la pantalla y el glifo no podrá ser menor a un único píxel. En este escenario, para una cantidad dada de datos a visualizar, no podrán obtenerse resultados mucho mejores que el obtenido aprovechando el máximo H y el mínimo P posibles.

El caso particular en que $P = 1$ se denominará *índice de distintos ubicados*. Este índice

indica qué porcentaje de datos distintos tiene el conjunto de datos una vez discretizados y ubicados en la visualización:

$$\phi \approx f(x, h, 1) = \frac{1}{1 + e^{a\ln(x)+b\ln(h)+c\ln(1)+d}} = \frac{1}{1 + e^{a\ln(x)+b\ln(h)+d}}$$

Luego, si X es la cantidad de datos a visualizar y L es el máximo tamaño posible de la visualización en función del tamaño de la pantalla, el índice de distintos ubicados ϕ establece una cota máxima del índice de visibilidad τ para ese conjunto de datos. Si el índice ϕ obtenido es muy cercano a 1, entonces es esperable que sea posible visualizar esos datos con una ventana de menor tamaño y/o un glifo de mayor tamaño y aún así obtener un índice τ que indique una visualización aceptable.

Suponiendo que se cuenta con un monitor cuya máxima resolución es de 1920×1080; a continuación se analizarán algunos casos en los que el *índice de visibilidad* y el *índice de distintos ubicados* pueden asistir al usuario en la visualización de diferentes conjuntos de datos:

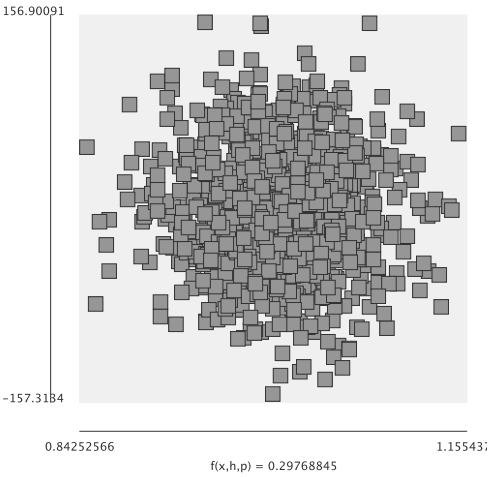
- El usuario tiene que visualizar un conjunto de 1058 datos:
  - Si decide visualizar dicho conjunto de datos en una ventana de 400×400 píxeles con un glifo de 16×16 píxeles (ver figura 5.5a), el usuario puede advertir que para estos parámetros el índice de visibilidad τ tiene un valor aproximado de 0,2976. Es decir, que solamente el 29 % de los glifos tendrían al menos un píxel no superpuesto con otro glifo.
  - Si decide visualizar dicho conjunto de datos en una ventana de 400×400 píxeles y establece la restricción de un valor mínimo aceptable para el índice de visibilidad τ de 0,9, el usuario puede advertir que el tamaño del glifo no debería ser mayor que 5×5 (ver figura 5.5b):

$$f(1058, 400, p) = \frac{1}{1 + e^{a\ln(1058)+b\ln(400)+c\ln(p)+d}} \geq 0{,}9$$

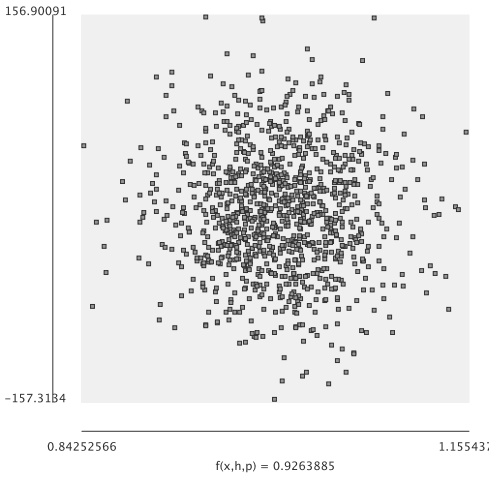
$$p \leq 5{,}60941$$

- El usuario tiene que visualizar un conjunto de 300 000 datos:
  - Si decide visualizar dicho conjunto de datos en una ventana de 400×400 píxeles, puede advertir que, aunque elija un glifo de 1 píxel, no lograría visualizaciones con un índice de visibilidad mayor a 0,03615.
  - Si para visualizar dicho conjunto de datos establece la restricción de un índice de visibilidad no menor a 0,9, podrá advertir que, aunque elija visualizar con glifos de 1 píxel, el tamaño de la visualización sería mayor que el espacio disponible en el monitor:

$$f(300\,000, h, 1) = \frac{1}{1 + e^{a\ln(300\,000)+b\ln(h)+c\ln(1)+d}} \geq 0{,}9$$



(a) 1058 datos visualizados en un *scatterplot* de 400 píxeles con glifos de 16×16 píxeles.



(b) 1058 datos visualizados en un *scatterplot* de 400 píxeles con glifos de 5×5 píxeles.

Figura 5.5: En el primer caso (figura (a)), el uso de la métrica puede advertir al usuario de que únicamente un 29 % de los glifos van a ser siempre visibles (independientemente del orden en que se dibujen). En el segundo caso (figura (b)), la métrica advierte al usuario de que para obtener un 90 % de glifos siempre visibles en una ventana de 400×400 se debe utilizar un glifo de tamaño 5×5 como máximo.

$$h \geq 2155{,}73$$

En aquellos casos en los cuales el índice de visibilidad no es favorable, incluso con el glifo más pequeño y la ventana más grande posibles, un sistema que soporte esta técnica debería desestimarla como potencialmente aceptable para el conjunto de datos en cuestión.

Al principio de este capítulo se habló de *guiar al usuario en el proceso de selección de los diferentes parámetros involucrados en la visualización*. La utilidad de las métricas crece si todas las técnicas están acompañadas de al menos una medida de escalabilidad visual y estas son empleadas por un sistema semiautomático con la capacidad de advertir al usuario de las diferentes decisiones que puedan degradar la visualización, y guiarlo en la elección de los parámetros que generen la mejor vista posible, así también como sugerirle técnicas alternativas (en base a sus propias medidas asociada) en los casos en los que la *mejor vista posible* con la técnica elegida no sea satisfactoria.

5.5. Conclusiones y trabajo futuro

En este capítulo se presentó una métrica que dada la cantidad de datos a visualizar, el tamaño de la ventana y el tamaño del glifo a utilizar, estima la cantidad de glifos que serán siempre visibles en ese caso particular. Esta métrica es útil a la hora de asistir al usuario en la elección de los parámetros más acertados para lograr una visualización

aceptable de los datos. En el análisis de esta métrica, sería deseable analizar y extender el modelo matemático a otras distribuciones de los datos.

Si bien se presentó una única métrica sobre los *scatterplots*, el objetivo a futuro es que el usuario (o el sistema semiautomático derivado) cuente con un abanico de métricas que le permitan medir diferentes características de la potencial visualización resultante, y de esta forma, disponer de más herramientas para elegir los parámetros más apropiados para visualizar un conjunto de datos con *scatterplots*.

Capítulo 6

Visualización de árboles

Se introducen los conceptos de visualización implícita y explícita de árboles. Además, se presentan los treemaps *como una técnica de visualización representativa de la visualización implícita de árboles. Finalmente, dentro de las técnicas explícitas de visualización de árboles, se propone el Gyrolayout, que es un* layout *hiperbólico 3D de árboles, generado en base a conceptos teóricos de análisis hiperbólico y a algoritmos de teselados discretos.*

6.1. Introducción

La visualización de árboles se encarga de representar gráficamente estructuras jerárquicas o *árboles*, que son estructuras presentes tanto en ciencias de la computación como en otras disciplinas. Estas estructuras jerárquicas pueden contener hasta tres tipos de información de interés:

1. La información estructural de la jerarquía (cómo se organizan los nodos).
2. La información asociada a cada nodo.
3. La información asociada a cada enlace.

Qué información es la que se desea visualizar condiciona la técnica de visualización apropiada para cada caso.

Schulz [Sch11] establece tres propiedades que caracterizan casi cualquier técnica de visualización de árboles; estas son:

- *Dimensionalidad*: 2D, 3D o híbrido.

  Distingue las visualizaciones según si son en el plano (2D) o en el espacio (3D). Una visualización de dimensionalidad híbrida puede ser una visualización en un espacio 2.5D o una composición de visualizaciones 2D y 3D.

- *Representación de los arcos*: explícito, implícito o híbrido.

  La representación de los arcos es explícita cuando se dibujan las conexiones entre los nodos. En una representación implícita, la posición relativa de los nodos determina

su conexión. Las variantes híbridas, utilizan ambos tipos de representación para diferentes partes del árbol.

- *Posicionamiento de los nodos*: radial, paralelo a ejes o libre.

  Un posicionamiento radial ubica a la raíz del árbol en el centro y a los hijos en un rango circular o esférico alrededor de esta. Por otro lado, un posicionamiento paralelo a ejes respeta las coordenadas cartesianas de la pantalla y mapea la altura de los nodos en un eje y el ancho del árbol en el otro eje. Los posicionamientos libres siguen estrategias más complejas.

En la actualidad existen muchos conjuntos de datos con elementos altamente relacionados entre sí. En particular, cuando las relaciones tienen una estructura jerárquica, el árbol es una poderosa abstracción y una gran cantidad de aplicaciones usan árboles para representar las relaciones entre datos. En este contexto es necesario proveer técnicas de visualización que puedan manejar grandes árboles.

Este capítulo se centra en dos técnicas de visualización de árboles, los *treemaps* como una técnica representativa de la visualización implícita de árboles y el Gyrolayout como una representativa de la visualización explícita. Inicialmente, se presentan los *treemaps* y su evolución a lo largo del tiempo para subsanar sus limitaciones. Luego, se presenta el Gyrolayout ([Urr+13]), que es un *layout* interactivo de árboles hiperbólico, basado en el *layout* presentado por T. Munzner [Mun97; Mun00]. El Gyrolayout usa el Teselado baricéntrico pesado de Voronoi y los espacios gyrovectoriales, que son una abstracción matemática natural para tratar con la geometría hiperbólica.

6.1.1. Limitaciones de las visualizaciones de árboles

Los árboles son un caso particular de los grafos, por lo tanto es esperable que la visualización de árboles tenga las mismas limitaciones que la de grafos presentadas en la sección 2.4.4:

1. *Sobrecarga.* El gráfico se sobrecarga de información y se torna visualmente confuso.

2. *Posicionamiento de los nodos.* La jerarquía define un orden parcial entre los elementos y la visualización debe no solo respetarlo sino también evidenciarlo.

3. *Tensión perceptual.* La relación padre–hijo es uno de los aspectos más importantes a visualizar en una jerarquía. Además, existen relaciones entre nodos como “hermanos”, “descendiente”, etc. que también pueden ser relaciones importantes a distinguir en la visualización.

6.1.2. Soluciones

Se han desarrollado diferentes técnicas que compensan las limitaciones de la visualización de árboles, por ejemplo:

- Los *treemaps*, *piecharts* y sus derivados son técnicas de visualización de árboles que representan de forma implícita los arcos, respetando el orden parcial existente. Además, la sobrecarga de información se evita ya que los nodos más profundos en el árbol tienden a cero y no son representados. Sin embargo, la tensión *perceptual* puede verse perjudicada ya que los nodos que se encuentran cerca en la visualización, están cerca en la estructura, sin embargo, la inversa no siempre es válida.
- Los árboles hiperbólicos, mantienen la representación explícita de los arcos que respeta el orden parcial y reduce la tensión *perceptual*. Además, logran evitar la sobrecarga ya que los nodos más profundos en el árbol también tienden a cero.

6.2. *Treemaps*

Los *treemaps* [JS91] son una técnica clásica de visualización de árboles; esta técnica permite mapear toda la información jerárquica a un rectángulo usando un modelo de rellenado de superficie; de esta forma se utiliza el 100 % de la superficie designada a la visualización (ver figura 6.1a). El uso eficiente del espacio permite visualizar grandes jerarquías completas (ver figura 6.1b). El principal objetivo de este diseño fue:

- Utilización eficiente del espacio.
- Interactividad.
- Comprensión.
- Estética en el dibujado y en la realimentación.

Los *treemaps* permiten visualizar tanto la estructura como el contenido de la jerarquía. Sin embargo, se desempeñan mejor cuando lo importante a visualizar es la jerarquía y el contenido de las hojas y además, el contenido de los nodos internos puede derivarse del contenido de sus hijos.

La creación de un *treemap* utiliza un algoritmo de particionamiento para dividir el rectángulo principal en pequeños rectángulos que representen los nodos. El algoritmo original (*Slice and Dice*) divide el rectángulo principal a lo ancho (verticalmente) en tantos rectángulos como hijos tuviera la raíz. Luego, cada uno de estos rectángulos obtenidos se divide a lo alto (horizontalmente) en función de la cantidad de hijos de cada uno. Este proceso se repite, alternando división vertical y horizontal, hasta llegar a las hojas del árbol. El área de cada hoja del árbol se corresponde con la cantidad de información que contiene. Este algoritmo puede resultar en rectángulos muy finos que confunden al usuario. En general, los rectángulos con relación de aspecto cercana a 1, es decir, lo más similares a un cuadrado, tienen varias ventajas:

- El espacio se usa más eficientemente ya que el perímetro (y por lo tanto el número de píxeles necesarios para representarlo) de un rectángulo de determinada área es menor cuando es un cuadrado.

- Los cuadrados son más fáciles de distinguir desde el punto de vista del usuario, mientras que los rectángulos angostos muy juntos generan errores de *aliasing*.
- La comparación de tamaños entre rectángulos es más sencilla cuando la relación de aspecto es similar.

Por lo tanto, se han propuesto algoritmos para balancear la relación de aspecto de los *treemaps*, y así evitar rectángulos muy finos. Sin embargo, esto introdujo un nuevo problema: algoritmos que no preservan el orden de los datos subyacentes y que no garantizan estabilidad, es decir, frente a pequeños cambios en los datos la representación cambia drásticamente. Hasta ahora se han desarrollado 7 algoritmos de particionamiento, que se resumen en la tabla 6.1.

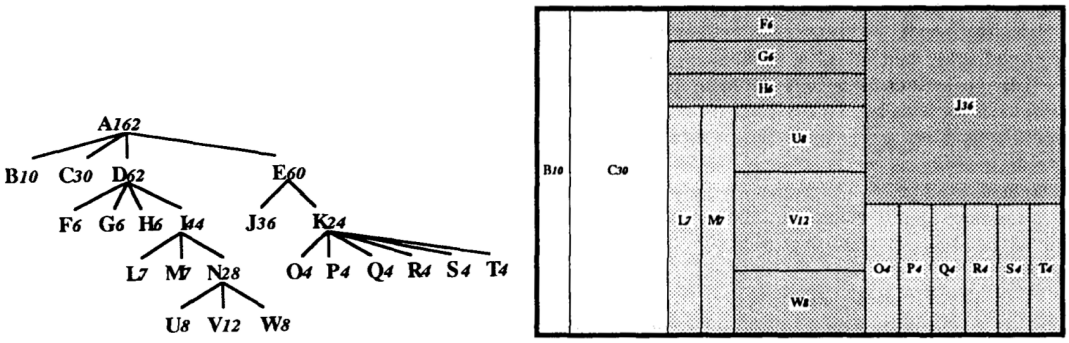
Algoritmo	Orden	Relación de aspecto	Estabilidad
Slice and Dice [JS91]	Ordenado	muy mala	estable
BinaryTree [Bru+99]	Orden parcial	media	estable
Squarified [Bru+99]	Desordenado	muy buena	inestable
Cluster treemaps [Wat99]	Desordenado	media	inestable
Ordered [Bed+02]	Orden parcial	media	inestable
Strip [Bed+02]	Ordenado	media	inestable
Mixed treemaps [Vli+06]	Ordenado	muy buena	estable

Tabla 6.1: Algoritmos de particionamiento para la creación de *treemaps* y sus características.

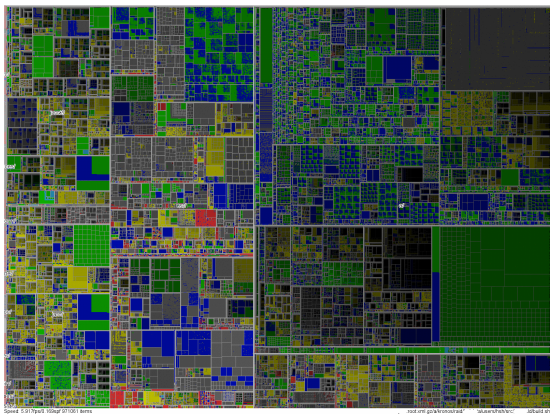
Hay casos en los que los *treemaps* no alcanzan para visualizar la estructura de los árboles. El peor caso es el de un árbol balanceado (los nodos internos tienen todos la misma cantidad de hijos y las hojas tienen todas el mismo peso) donde el *treemap* degenera a una grilla regular.

Los *treemaps* son una visualización compacta, que logra ser efectiva para mostrar el tamaño de los nodos de la jerarquía ([Bru+99; Vli+06]). Esto la ha convertido en una técnica de visualización ampliamente usada en muchas aplicaciones. A lo largo de los años se han desarrollados diferentes variantes:

- Los *Cushion treemaps* [WW99] agregan sombreado para destacar mejor la estructura del árbol visualizado (ver figura 6.2a).
- Los *Treemaps circulares – Pebbles* [Wet03] son una alternativa visualmente atractiva, además de que el anidamiento es notablemente visible; sin embargo, el uso del espacio no es eficiente (ver figura 6.2b).
- Los *treemaps*, en principio, están limitados a formas rectangulares, lo que genera problema con la relación de aspecto de los rectángulos y con identificar la estructura de la jerarquía. Los *Voronoi treemaps* [BD05a] permiten dividir cualquier polígono (triángulos, círculos, etc.) y generar subdivisiones con formas arbitrarias (ver figura 6.2c).



(a)



(b) *Treemap* con un millón de ítems. Imagen tomada de `http://www.cs.umd.edu/hcil/VisuMillion/`.

Figura 6.1: Ejemplos de *treemaps*. En la figura (a) se muestra cómo se subdivide el espacio para representar una jerarquía con sus nodos en un espacio reducido. En la figura (b) se muestra cómo la técnica es capaz de visualizar grandes jerarquías de datos.

- Los *treemaps generalizados* [Vli+06] se presentan como un nuevo algoritmo de subdivisión que permite crear *treemaps* con la apariencia de gráficos de negocios y se introducen como una alternativa para visualizar datos empresariales (ver figura 6.2d).
- *Zoomable treemaps* [BL07] mejoran el *treemap* tradicional logrando que sea multiescala y, por lo tanto, apropiado para explorar grandes árboles (ver figura 6.2e).
- Los *3D–treemaps* [Lig+09] representan cada nodo de la jerarquía con un prisma, y permiten mapear diferentes propiedades de cada nodo a las propiedades gráficas del prisma que lo representa: tamaño, color y altura (ver figura 6.3a).
- *Treecube* [Tan+03] es una técnica de visualización de información jerárquica que se basa en el anidamiento de cubos (ver figura 6.3b), como una extensión a 3D de los *treemaps*.

- *Strata treemaps* [Cho+11] es una adaptación 3D de los *treemaps* que combina una representación esférica con un Teselado baricéntrico pesado de Voronoi (ver figura 6.3c).

6.3. El Gyrolayout

Agregar la tercera dimensión en la representación visual de los árboles genera más espacio y más alternativas para dibujar nodos que en el espacio 2D. Esta característica hace que los *layouts* 3D de árboles sean apropiados para visualizar grandes árboles.

Las técnicas con varios niveles de detalle (LOD) se basan en simplificar los detalles de objetos complejos o de grupos de objetos. Esta simplificación puede lograrse sustituyendo un objeto complejo o un grupo de objetos por un objeto representativo más simple. El nivel de simplificación depende de cuán significativo es el nuevo objeto en comparación con el objeto o el grupo de objetos reemplazado. En general, el LOD facilita la exploración del conjunto de datos disminuyendo la carga visual.

El Gyrolayout acepta varios niveles de detalle y puede mostrar en el orden de 10^5 nodos. La geometría hiperbólica es regulada por los espacios gyrovectoriales, de la misma forma en que los espacios vectoriales regulan la geometría euclídea; esto permite contar con un marco controlado para realizar las transformaciones geométricas. El punto más destacable es cómo el análisis teórico de la geometría hiperbólica y los algoritmos de teselado interactúan para lograr una visualización interactiva de grandes árboles.

6.3.1. Antecedentes

Una estrategia común usada para ganar espacio para visualizar árboles y grafos es diseñar visualizaciones 3D en vez de visualizaciones 2D. Generalmente, los algoritmos 3D son adaptaciones de los 2D. Sin embargo, los resultados no son siempre ventajosos: por ejemplo, la oclusión entre objetos y la navegación del espacio tridimensional, son problemas que aparecen debido a la dimensión adicional.

Algunos *layouts* 3D de árboles son generalizaciones de los *layouts* 2D; por ejemplo, el *layout* Esférico [Lar06] es una generalización del *layout* radial. El *Cone Tree* [Rob+91] es un *layout* desarrollado directamente en 3D. Además, las representaciones esféricas han sido exploradas para lograr mejorar las visualización de árboles y grafos ([DV09], [Cho+11], [Sch+11], [BM12]). En particular, *Strata Treemaps* [Cho+11] combina la representación esférica con un Teselado baricéntrico pesado de Voronoi (en cuanto a lo que se conoce el algoritmo no fue publicado) y [DV09] combina la representación esférica con técnicas de *sphere-packing*.

Otra estrategia para ganar aún más espacio, es ubicar el árbol en el espacio hiperbólico: el *layout* H3 [Mun97] es una generalización de las visualizaciones de árboles que usan proyecciones de geometría hiperbólica en el plano [Lam+95]. Walrus [CAI05] es una implementación para visualizar árboles en el espacio hiperbólico 3D.

El espacio hiperbólico es apropiado para la visualización de grandes árboles: tanto el espacio disponible como los árboles crecen exponencialmente. Y, junto con alguna estrategia multirresolución, sería capaz de ubicar árboles aún más grandes.

6.3.2. Características generales del *layout*

Se diseñó e implementó un *layout* que acepta varios niveles de detalle (LOD) y es apropiado para representar grandes árboles (del orden de los 10^5 nodos) en el espacio hiperbólico tridimensional. Este *layout* permite visualizar árboles con diferentes niveles de detalle en un volumen cerrado (una bola) embebido en el espacio euclídeo tridimensional. La geometría hiperbólica es regulada por los espacios gyrovectoriales que se detallan en el apéndice A, de la misma forma en que los espacios vectoriales regulan la geometría euclídea; esto permite contar con un marco controlado para realizar las transformaciones geométricas.

En el *layout* se utilizan dos estrategias diferentes para ubicar los nodos. Primero, se ubica la raíz dentro de la bola unitaria (por ejemplo, en el centro de la esfera), y los hijos se ubican ocupando el mayor espacio posible alrededor de la raíz; para lograr esto se utiliza el Teselado esférico, baricéntrico y pesado de Voronoi ([Lar+09; Urr+13]), que se detalla en el apéndice B. Para ubicar en el espacio los nodos de profundidad mayor o igual a 2 (es decir, los descendientes de los hijos de la raíz) el *layout* se basa en el esquema presentado en [Mun97]: los hijos de un nodo n se ubican en una superficie semiesférica centrada en el nodo; el hemisferio apunta en la dirección del padre de n hacia n (ver figura 6.4).

El algoritmo consta de tres pasos importantes:

A) Un recorrido ascendente que calcula el radio de los hemisferios donde se ubicarán los hijos de cada nodo (exceptuando la raíz).

B) Un Teselado esférico, baricéntrico y pesado de Voronoi que distribuye los hijos de la raíz en sus posiciones finales sobre la superficie de una esfera.

C) Un recorrido descendente que ubica los nodos restantes (los descendientes de los hijos de la raíz).

A continuación se describen en detalle cada uno de estos tres pasos. Luego, se presenta el algoritmo que genera el Gyrolayout, basado en los tres algoritmos previamente descriptos. Por último, se presenta la estrategia para lograr diferentes niveles de detalle.

Calculando los radios de los hemisferios

El *recorrido ascendente* (esquematizado en el algoritmo 1) calcula el radio del hemisferio necesario para ubicar los hijos de cada nodo. Comienza ordenando los hijos de cada nodo en un disco. Luego, por cada disco, calcula el radio de un hemisferio con la misma área que el disco, donde se proyectarán los nodos.

Sea n un nodo y $h_1, h_2, \ldots, h_m$ sus m hijos. Inicialmente, el recorrido ascendente calcula recursivamente, para cada $h_1, h_2, \ldots, h_m$, el tamaño del disco necesario para ubicar los

hijos (paso 1.1), que es el tamaño necesario para ubicar todos los *nietos* de n. Luego, se ordenan en anillos concéntricos (pasos 1.3–1.7), comenzando por un círculo interno concéntrico. Los nodos deben ordenarse de tal forma que aquellos con discos mayores ocupen los anillos centrales y, a medida que el tamaño del disco disminuye, los nodos se van ubicando en los anillos más alejados del centro (ver figura 6.5). Para las hojas se asocia un tamaño (radio) equivalente al radio de la esfera que delimita el volumen ocupado por el elemento visual que representa la hoja. Dado que cada anillo mantiene información de la distancia δ (ver figura 6.5) y el ángulo ocupado α, es posible determinar si hay suficiente espacio para ubicar un nodo en un anillo (paso 1.5 del algoritmo 1), es decir:

1: Sea $\mathcal{R}$ un anillo de distancia δ y ángulo ocupado α.
2: Sea n un nodo de radio r.
3: Hay suficiente lugar para n en $\mathcal{R}$ si $2\pi - \alpha \geq \sin^{-1} r/\delta$.

Algoritmo 1 Distribuir los hijos del nodo n en anillos

▷ *Recorrido ascendente*

Entrada: Un nodo n
Salida: Distribución en anillos de los hijos de n

1.1: **para cada** hijo h de n
1.2: Distribuir en anillos los hijos de h. ▷ *Cálculo recursivo de los anillos de los hijos* ▷ *Acomodar los nodos en anillos concéntricos*
1.3: Sea $i = 0$. ▷ *Círculo interior concéntrico*
1.4: **para cada** hijo h de n ▷ *Los nodos se recorren en orden descendente según el tamaño del disco*
1.5: **si** no hay suficiente lugar para poner h en el anillo i **entonces**
1.6: Incrementar i en 1. ▷ *Saltar al siguiente anillo alejándose del centro*
1.7: Ubicar h en el anillo i
1.8: Sea R_{out} la suma de los anchos de todos los anillos necesarios
1.9: Sea $R_s = \sinh^{-1} \sqrt{\cosh(R_{out}) - 1}$ el radio del hemisferio correspondiente al nodo n

Una vez que todos los hijos están acomodados, el algoritmo calcula el tamaño (radio) de todo el círculo (paso 1.8) y finalmente, calcula el tamaño (radio) de un hemisferio con la misma área (paso 1.9). La figura 6.6 muestra dos vista de un hemisferio que contiene solamente hojas. En el espacio hiperbólico, el área de un círculo de radio r y la superficie de una esfera de radio r son[1], respectivamente:

área del círculo	$4\pi \sinh^2(\frac{r}{2})$
área de la esfera	$4\pi \sinh^2(r)$

Luego, la superficie de un hemisferio de radio r es $2\pi \sinh^2(r)$. En este algoritmo es necesario encontrar el radio R_s de un hemisferio con la misma área que un círculo de

[1] Para más detalles históricos y del desarrollo de estas fórmulas referirse a [Bon12].

radio R_{out}. Despejando R_s de la siguiente igualdad

$$4\pi \sinh^2\left(\frac{R_{out}}{2}\right) = 2\pi \sinh^2(R_s)$$

y simplificando, se obtiene (paso 1.9):

$$R_s = \sinh^{-1}\sqrt{2\sinh^2\left(\frac{R_{out}}{2}\right)} = \sinh^{-1}\sqrt{\cosh(R_{out}) - 1}.$$

Al finalizar el algoritmo, se ha calculado el radio del hemisferio necesario para ubicar los hijos del nodo n y los radios de los hemisferios necesarios para los hijos de cada uno de los descendientes de n.

Ubicando la raíz y sus hijos

La raíz puede ubicarse en cualquier lugar dentro de la esfera; por ejemplo, se puede elegir el centro de la bola como una buena posición inicial. Luego, los nodos deben distribuirse en el espacio disponible alrededor de la raíz. La estrategia elegida fue distribuir los hijos de la raíz r en la superficie de una bola imaginaria centrada en r, usando el Teselado esférico, baricéntrico y pesado de Voronoi (WSCVT) presentado en la sección B.

El peso de cada nodo indica cuál es el área de la esfera que va a necesitar el nodo para ubicar a sus hijos: si el nodo es una hoja, necesitará mucho menos espacio que otro nodo con varios hijos. Por lo tanto, el peso de cada nodo debe tener en cuenta el área necesaria para ubicar los descendientes.

Ubicando los nodos restantes

El *recorrido descendente* (algoritmo 2) calcula las posiciones finales en el espacio hiperbólico de los hijos del nodo n. Para lograr esto, el algoritmo necesita las gyroposiciones finales $\mathbf{p}_n$ y $\mathbf{p}_f$ de n y su nodo padre f, respectivamente, además de los anillos asociados a n obtenidos a través el algoritmo 1, previamente descripto. El caso particular de calcular la posición de la raíz y sus hijos (estas últimas son datos de entrada de este algoritmo) se detalla en el algoritmo 8 del apéndice B. El *recorrido descendente* pasa por cada uno de los nodos de cada anillo (paso 2.7) y calcula la gyroposición de los nodos. Los anillos son recorridos desde el anillo central hacia el perimetral (paso 2.3).

Gyrolayout. Integrando todos los algoritmos

A continuación se describe cómo los algoritmos previos se ensamblan para obtener el Gyrolayout, y se dan los detalles de los tres pasos principales (A, B y C) presentados en la introducción de esta sección (algoritmo 3).

El algoritmo tiene como datos de entrada el árbol a ser visualizado y la posición del nodo raíz en el espacio hiperbólico (una gyroposición) y crea la visualización final del árbol. El primer paso es distribuir todos los nodos en anillos (paso 3.2). Si la raíz tiene

Algoritmo 2 Calcular la gyroposición de los hijos del nodo n ▷ *Recorrido descendente*
Entrada: Un nodo n
La gyroposición $\mathbf{p}_n$ de n
El gyrovector normalizado $\mathbf{d}_n$ que apunta en la dirección del hemisferio de n
Salida: Las gyroposiciones de los hijos de n

2.1: Sea r_n el radio del hemisferio que contiene los hijos de n.
2.2: Sea $\langle \mathbf{b}_1, \mathbf{b}_2, \mathbf{b}_3 \rangle$ una base ortonormal tal que $\mathbf{b}_1 = \mathbf{d}_n$
2.3: **para cada** anillo $\mathcal{A}$ donde se ubican los hijos de n
2.4: Sea $\delta_{\mathcal{A}}$ la distancia del medio del anillo al centro del círculo que lo contiene. ▷ *Ver figura 6.5*
2.5: Inicializar ϕ en 0
2.6: Se define la rotación $R_{\mathbf{b}_3,\beta_{\mathcal{A}}}$, donde $\beta_{\mathcal{A}} = \delta_{\mathcal{A}}/r_n$
2.7: **para cada** nodo m dentro del anillo $\mathcal{A}$
2.8: Sea $\mathbf{d}_m = \mathbf{b}_1$ un gyrovector normalizado apuntando en la dirección del hemisferio de m.
2.9: $R_{\mathbf{b}_3,\beta_{\mathcal{A}}}(\mathbf{d}_m)$ ▷ *Rotatar* $\mathbf{d}_m$
2.10: Inicializar α_m con $2\sin^{-1} r_m/\delta_{\mathcal{A}}$
2.11: Incrementar ϕ en $\alpha_m/2$
2.12: Se define la rotación $R_{\mathbf{b}_1,\phi}$
2.13: $R_{\mathbf{b}_1,\phi}(\mathbf{d}_m)$ ▷ *Rotatar* $\mathbf{d}_m$
2.14: Incrementar ϕ en $\alpha_m/2$
2.15: Asignar $\mathbf{p}_m = \mathbf{p}_n \oplus (r_n \otimes \mathbf{d}_m)$ ▷ $T_{r_n \otimes \mathbf{d}_m}(p_n)$
2.16: *Calcular la gyroposición de los hijos del nodo* m $(m, \mathbf{p}_m, \mathbf{d}_m)$

suficiente cantidad de hijos (por ejemplo, más de 4 hijos), el algoritmo considera el peso inicial de esos hijos como el radio del disco donde se ubican los anillos concéntricos con centro en el nodo (step 3.4) y calcula el WSCVT correspondiente (paso 3.5). Luego de esto, el algoritmo calcula el radio ρ de una esfera con suficiente área para contener los discos de todos los hijos de la raíz (paso 3.6). A continuación, para cada hijo i de la raíz, se proyecta la gyroposición de i (calculada por el WSCVT) sobre la superficie de la esfera de radio ρ y finalmente se calcula el gyrovector que apunta en dirección hacia afuera (paso 3.8 and 3.9). A partir de esos valores, el algoritmo calcula la gyroposición de todos los descendientes de cada hijo de la raíz (paso 3.10).

Si la raíz tiene muy pocos hijos (por ejemplo, como mucho 4 hijos), no se justifica calcular una distribución que cubra toda la superficie de la esfera ya que esta distribución va a ser muy rala. El algoritmo asume una dirección preestablecida y acomoda los hijos de la raíz también en un hemisferio (pasos 3.12 y 3.13).

Gyrolayout y los diferentes niveles de detalle

A medida que la posición de un nodo tienda a la superficie de la bola de Klein, su elemento visual se hará más pequeño, ya que la superficie de la bola representa el infinito del espacio hiperbólico. Además, a medida que el nodo se encuentre más profundo en el

Algoritmo 3 Visualizar el árbol $\mathcal{T}$ ▷ *Juntando todos lo algoritmos*

Entrada: Un árbol $\mathcal{T}$

Una gyroposición $\mathbf{O}$ que será la posición de la raíz dentro de la bola, puede ser $\{0, 0, 0\}$

3.1: Sea r el nodo raíz de $\mathcal{T}$
3.2: *Distribuir los hijos del nodo r en anillos* ▷ *Algoritmo 1*
3.3: **si** r tiene más de 4 hijos **entonces**
3.4: Sea $W = \{w_1, \ldots, w_n\}$ el peso de los hijos de r, donde w_i es el peso del hijo i
3.5: Sea $P = \{\mathbf{p}_1, \ldots, \mathbf{p}_n\}$ el resultado de WSCVT(W) ▷ *Algoritmo 8*
3.6: $\rho = \sqrt{\frac{\sum \pi w_i}{4\pi}} = \frac{\sqrt{\sum w_i}}{2}$
3.7: **para cada** hijo i de r
3.8: Asociar a i la gyroposición $\mathbf{g}_i = \rho \otimes \frac{\mathbf{p}_i}{\|\mathbf{p}_i\|}$
3.9: Sea $\mathbf{d}_i = \frac{\ominus \mathbf{O} \oplus \mathbf{g}_i}{\|\ominus \mathbf{O} \oplus \mathbf{g}_i\|}$ un gyrovector que apunta en la dirección del hemisferio de i
3.10: *Calcular la gyroposición de los hijos del nodo $i(i, \mathbf{g}_i, \mathbf{d}_i)$* ▷ *Algoritmo 2*
3.11: **si no**
3.12: Sea $\mathbf{d}$ una dirección, por ejemplo $\{\tanh 1, 0, 0\}$
3.13: *Calcular la gyroposición de los hijos del nodo $r(r, \mathbf{O}, \mathbf{d})$* ▷ *Algoritmo 2*
3.14: Visualizar $\mathcal{T}$

árbol, su posición estará más lejos de la raíz y más cerca de la superficie de la bola. Luego, a medida que un nodo esté más profundo en el árbol, su elemento visual se empequeñecerá y tenderá a desaparecer.

Teniendo esto en cuenta, un criterio apropiado para establecer un nivel de detalle podría ser cuán apreciables son los detalles de un manojo de nodos: se pueden podar aquellos subárboles para los cuales el elemento visual de los nodos sea muy pequeño.

Todos los nodos son representados por tetraedros debido a su reducido número de caras. El tetraedro permite interactuar con el nodo y agregarle a la visualización propiedades del nodo. Un subárbol que ha sido recortado se marca con un cono para evidenciar la existencia de datos ocultos. El cono es un elemento visual diferente y significativo (basado en los árboles fusionados de [CK95]) que representa la raíz de árbol podado (ver figura 6.7).

Algoritmo 4 Colapsar el subárbol de raíz n

Entrada: Un nodo n
Salida: Un cono que representa a n y a los descendientes de n

4.1: Sea $\mathbf{p}_n$ la gyroposición de n
4.2: Sea $\mathbf{p}_m$ la gyroposición del nodo en el anillo central de n
4.3: Sea $\mathbf{p}_f$ la gyroposición del nodo padre de n
4.4: Sea $\mathbf{p}_n$ el vértice del cono
4.5: Sea $\mathbf{d} = \ominus\mathbf{p}_f \oplus \mathbf{p}_n$ la dirección del cono
4.6: Sea $h = \|\mathbf{p}_n \ominus \mathbf{p}_m\|$ la altura del cono
4.7: Sea r_n el radio del hemisferio de n
4.8: **retornar** Un cono de vértice $\mathbf{p}_n$, altura h, orientado en la dirección de $\mathbf{d}$ y base de radio r_n.

En el algoritmo 4 se esquematiza cómo calcular el tamaño del cono que representará al subárbol podado.

6.3.3. Visualización de árboles utilizando el Gyrolayout

La visualización de árboles no implica únicamente ubicar nodos y aristas en el espacio, sino también decidir qué atributos del conjunto de datos se representarán visualmente y cómo se mapearán, además de qué interacciones deberán proveerse. En general, las aplicaciones de visualización de árboles difieren en alguno de los siguientes aspectos:

- Usan diferente diagramado para ubicar nodos y aristas.
- Usan diferente mapeo visual para los atributos.
- Proveen un conjunto diferente de interacciones.

Esta propuesta usa el Gyrolayout para ubicar nodos y aristas. La representación de los atributos y las interacciones son fuertemente dependientes del dominio de aplicación. A continuación se discutirán algunos aspectos clave sobre el mapeo visual y las interacciones.

Mapeo Visual

Al momento de determinar un mapeo visual para nodos y aristas es necesario tener en cuenta ciertos factores para mejorar la visualización y alcanzar un mejor entendimiento de los datos que representa el árbol.

- Las aristas del árbol se colorean de acuerdo a la profundidad del nodo destino. El color toma valores de un degradé de tonos, donde un extremo corresponde al color de las aristas con origen en la raíz del árbol y el otro extremo, a las aristas con destino en un nodo con máxima profundidad (ver figura 6.11).
- Los nodos de un árbol se pueden colorear de acuerdo a su profundidad dentro del árbol o en función de la semántica del nodo dentro del conjunto de datos.
- La porción del árbol ubicada en el hemisferio opuesto al observador se muestra menos saturado, para dar al usuario información adicional sobre qué objetos están cerca y qué objetos están lejos.

Interacciones

Herman et al. [Her+98] describen cuál sería el conjunto de interacciones para ayudar al usuario en la navegación de grandes árboles 2D.

1. *Zoom y paneo* son las bases de las interacciones de navegación.
2. *Foco+contexto* a través del lente *ojo de pez*.
3. *Pistas de complejidad* para dar al usuario información que indique dónde están los subárboles interesantes, como por ejemplo, en qué dirección el árbol crece a través de números de Strahler.
4. *Plegado y desplegado* para ocultar subárboles específicos; sólo se deja la raíz como representante del subárbol completo.

Todas estas interacciones básicas se encuentran soportadas por el Gyrolayout. Teniendo en cuenta la naturaleza 3D, las interacciones son adaptadas a este contexto ([Car+99]):

1. El *zoom* y el *paneo* en representaciones 3D se corresponden con la rotación y la translación de la cámara, lo que hace posible al observador interactuar con el espacio hiperbólico sin cambiar la representación visual.
2. El *foco+contexto* es intrínseco a la representación elegida del espacio hiperbólico, donde los nodos cerca del centro de la bola de Klein se muestran más grandes y más detallados que aquellos que se encuentran cerca de la superficie de la bola.
3. Las *pistas de complejidad* se agregan a los nodos y son visibles a través de zoom semántico.

4. El *plegado* y el *desplegado* de los nodos son intrínsecos de la técnica de nivel de detalle: oculta algunos subárboles en función de algún criterio.

Además, se diseñó un conjunto adicional de interacciones para complementar el conjunto básico:

- *Zoom semántico en nodos y cono.* El zoom semántico posibilita el mostrado bajo demanda de información adicional sobre un nodo: cuando se cliquea un nodo se muestra un *tooltip* que informa, por ejemplo, la ruta absoluta de un archivo si el árbol fuera una jerarquía de directorios (ver figura 6.9b). En el caso de los nodos, el *tooltip* podría informar acerca de la altura del subárbol escondido (ver figura 6.9c).

- *Resaltado de caminos.* Cuando se cliquea un nodo, se resalta el camino desde ese nodo hasta la raíz.

- *Traslación del árbol en el espacio hiperbólico.* Además de la traslación de la cámara, que no cambia la representación hiperbólica del árbol, se permite trasladar el árbol en el espacio hiperbólico. Esta traslación se logra trasladando (ver *Traslación* en la sección A) todos los puntos desde su posición actual a la posición original (la raíz del árbol en el origen) y luego trasladando el nodo elegido al centro de la bola (ver figura 6.8). A pesar de que esta interacción resulta en posiciones hiperbólicas diferentes para cada nodo, y por lo tanto, altera la representación del árbol, resulta interesante ya que permite cambiar el punto de interés de la visualización.

Implementación

Se implementó un prototipo para visualizar árboles utilizando el Gyrolayout. Este fue implementado en JAVA usando la librería VTK. El prototipo provee las interacciones básicas implementadas por VTK: rotación y traslación de la cámara (ver figura 6.11), así como también las interacciones descriptas en la Sección 6.3.3.

Para poder tratar con grandes árboles y mejorar la interactividad, se tuvieron en cuenta algunas consideraciones durante la implementación para así hacer más eficiente el proceso de *rendering*:

- los nodos se representan con un símplex (un tetraedro).

- las aristas se representan con líneas y no tubos.

- los nodos no se muestran durante las transformaciones de la cámara.

A los fines de la implementación, se adaptó la clase `vtkGlyph3D` para realizar todas las transformaciones como gyrotransformaciones (esto es, las translaciones de acuerdo a la ley de adición del gyroparalelogramo, como muestra la figura A.1b).

Cantidad de nodos	Gyrolayout	H3	Walrus	
			Simple precisión	Doble precisión
2196	00 : 00,574	instantáneo	00 : 00,057	00 : 02,079
4444	00 : 01,575	un parpadeo	00 : 00,150	00 : 03,627
4994	00 : 01,790	un parpadeo	00 : 00,054	00 : 03,594
83845	00 : 15,219	4/5 seg.	00 : 00,555	01 : 03,578
138543	00 : 24,996	5/6 seg.	00 : 00,457	01 : 37,615
193331	00 : 34,863	8 seg.	00 : 01,722	02 : 19,130

Tabla 6.2: Comparación de Gyrolayout con H3 y Walrus. La comparación se desarrolló teniendo en cuenta cuánto tiempo le tomaba a cada aplicación visualizar árboles con diferentes cantidades de nodos (incluyendo el tiempo de preprocesamiento y *rendering*). Notar que no hay tiempos especificados para H3, ya que la aplicación no proveía tal información. Además, se pudo testear Walrus con doble y simple precisión, mientras que Gyrolayout usa sólo doble precisión.

Comparación con otras implementaciones 3D de árboles hiperbólicos

En esta sección se presenta una comparación (tabla 6.2) entre el Gyrolayout y otras alternativas hiperbólicas 3D desarrolladas: H3 [Mun97] y Walrus [CAI05]. La comparación se llevó a cabo en un procesador Core2Duo @2,66GHz con 4GB de RAM con un sistema operativo Debian GNU/Linux.

Mientras que Gyrolayout y Walrus fueron ambos implementados en Java, Gyrolayout usa VTK para las representaciones 3D y Walrus usa Java3D. H3 fue implementado en C++ y OpenGL y se testeó la versión para Windows a través de Wine[2] sobre la misma plataforma Linux. La comparación consiste en medir cuánto tarda cada aplicación en visualizar árboles de diferente cantidad de nodos (incluyendo el preprocesamiento y el tiempo de *rendering*). La comparación (tabla 6.2) muestra que los tiempos medidos para Gyrolayout están entre los de H3 y Walrus (con simple y doble precisión[3]). Esto es importante ya que Gyrolayout no sólo fue implementado con doble precisión, sino que además, permite las interacciones necesarias en el caso de representaciones a diferentes niveles de detalle como lo son las de plegar y desplegar nodos.

Además, H3 no permite rotar la bola que representa el espacio hiperbólico o hacer zoom in/out de la bola para ver más detalles. En su lugar, permite trasladar el árbol dentro del espacio hiperbólico; de esta forma logra una vista más detallada del centro de la visualización pero modifica la representación del árbol perdiendo el contexto (ver figura 6.10). En contraste con H3, Gyrolayout provee más detalles usando, además, paneo, *zoom* y rotación de la esfera (ver figura 6.11).

Con respecto a la ubicación de los nodos del árbol, tanto Walrus como H3 los ubican de forma que ocupan sólo un hemisferio (ver figura 6.12). Sin embargo, el *layout* presentado en este capítulo tiende a distribuir los nodos sobre toda la superficie de la esfera (ver figura 6.11(a)).

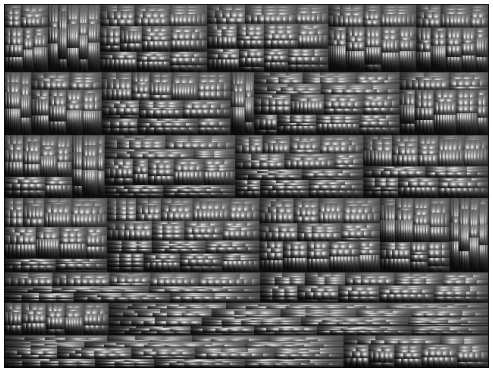
[2] `www.winehq.org`

[3] Walrus usa el paquete `mpfun` para poder manejar múltiples precisiones; esto reduce considerablemente el rendimiento al trabajar con doble precisión

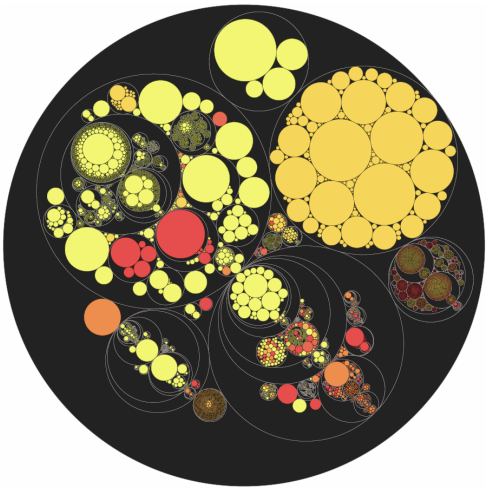
6.3.4. Conclusiones y trabajo futuro

Se desarrolló un diagramado de árboles basado en Teselados de Voronoi Centróideas y los espacios gyrovectoriales de Einstein que soporta diferentes niveles de detalle (ver figura 6.13). La principal fortaleza de este *layout* es que, debido a estar ubicado en el espacio hiperbólico y a soportar diferentes niveles de detalle, es realmente apropiado para visualizar grandes árboles (ver figura 6.14); se debe remarcar que es posible representar un árbol completo dentro de los límites de una esfera convencional en 3D. El análisis hiperbólico y los algoritmos de teselados discretos interactúan en este *layout* para lograr una visualización efectiva de grandes árboles.

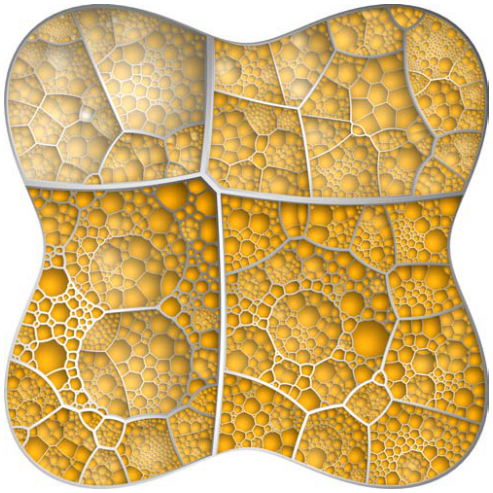
Como trabajo futuro se planea mejorar el *layout* en términos de complejidad visual y agregar interacciones adicionales, además de llevar a cabo test de usabilidad para validarlo. La meta final es adaptar el *layout* y la estrategia de niveles de detalle para soportar no solo grandes árboles sino también grandes grafos.

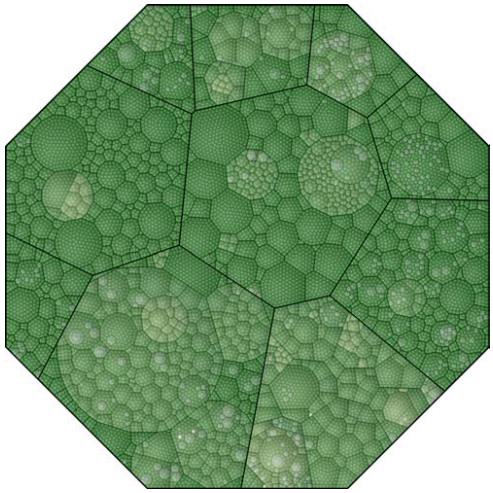


(a) *Cushion treemaps* (Página 74)



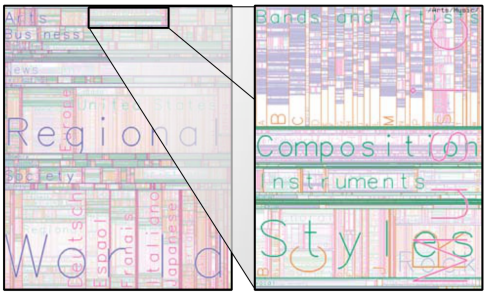
(b) *Treemaps* circulares – Pebbles (Página 74)





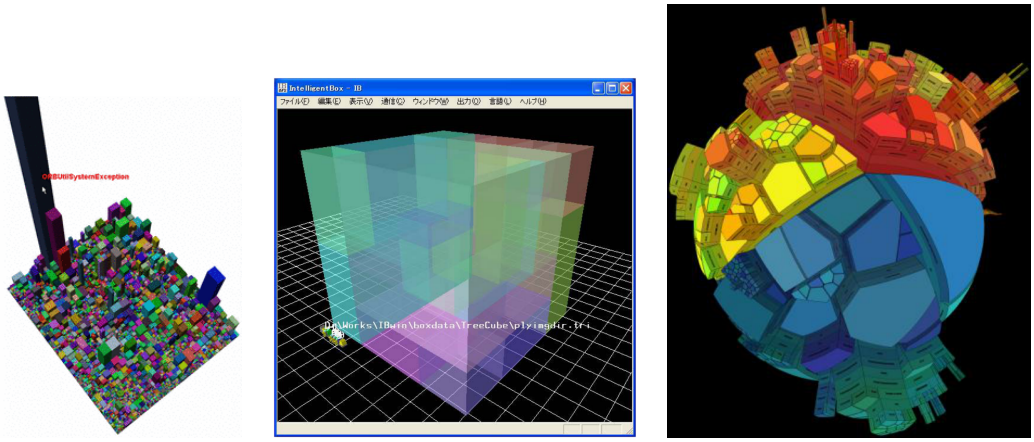
(c) Voronoi *treemap*. Dos alternativas diferentes para calcular la distancia pesada a los baricentros. (Página 74)

(d) *Treemaps* generalizados (Página 75)



(e) *Zoomable Treemap* (Página 75)

Figura 6.2: Adaptaciones 2D de los *treemaps*.



(a) 3D–*treemap* (Página 75) **(b)** Treecube (Página 75) **(c)** Strata *treemap* (Página 76)

Figura 6.3: Adaptaciones 3D de los *treemaps*.

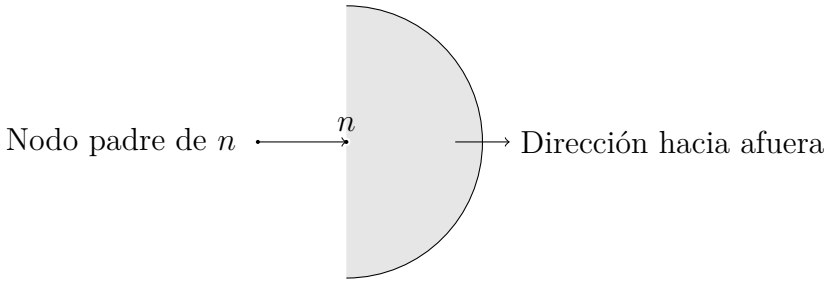


Figura 6.4: El hemisferio donde se ubican los hijos del nodo n está centrado en n y apunta en dirección del padre de n hacia n. (Página 77)

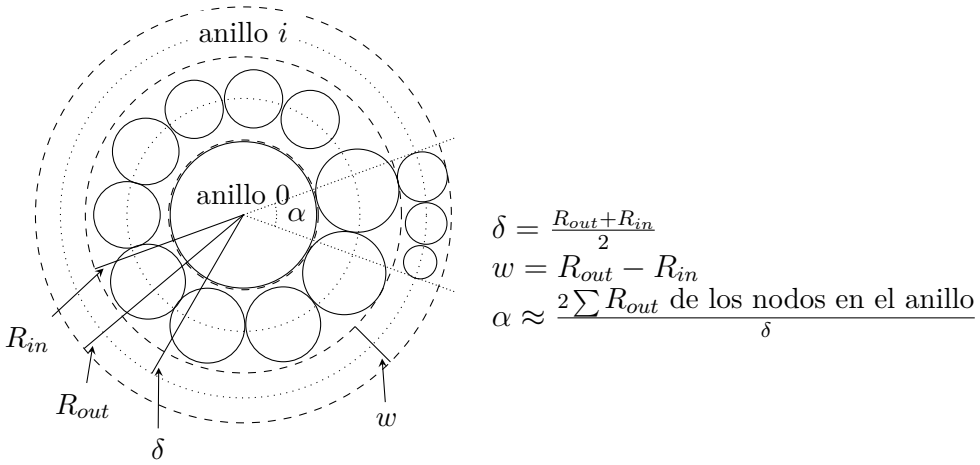
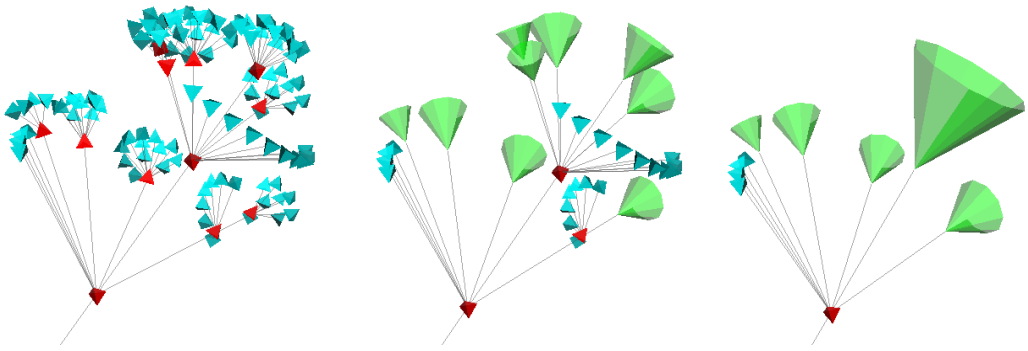


Figura 6.5: Anillos concéntricos de nodos. R_{out} y R_{in} son el radio exterior e interior del anillo i. δ es la distancia desde el centro del círculo al medio del anillo, w es el ancho del anillo, que se corresponde con el diámetro del nodo "más grande" ubicado en el anillo y α representa una aproximación del valor real del ángulo ocupado dentro del anillo. (Página 78)

(a) Vista superior **(b)** Vista lateral

Figura 6.6: Dos vistas diferentes de un hemisferio que contiene únicamente hojas. (Página 78)



(a) **(b)** **(c)**

Figura 6.7: Diferentes niveles de detalle. Las tres figuras muestran el mismo subárbol con diferentes niveles de detalle. La figura (a) muestra el detalle completo. En la figura (b) se han colapsado aquellos nodos cuyos hijos son todos hojas. En la figura (c) se han colapsado los nodos cuyos hijos son hojas o han sido colapsados en (b). (Página 82)

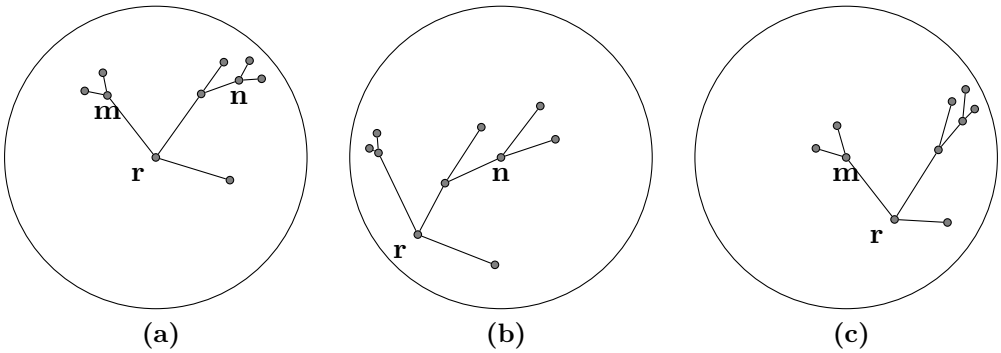
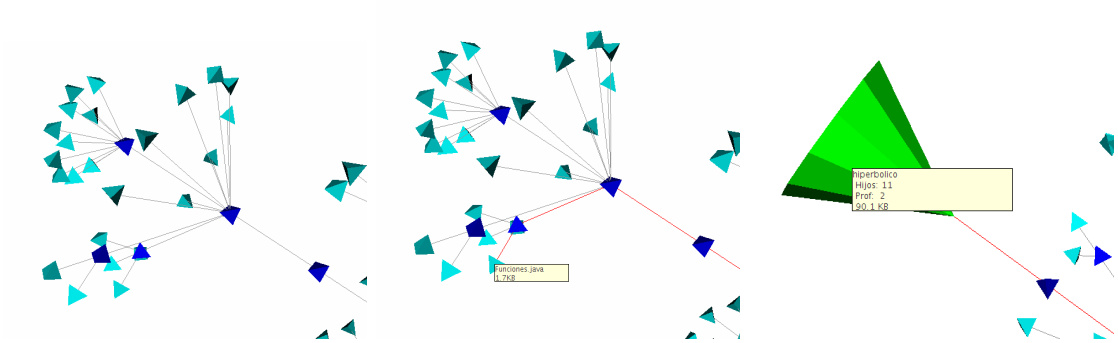


Figura 6.8: Un ejemplo de traslación del *layout*. La figura (a) muestra el diagramado original del árbol, en el cual la raíz del árbol se encuentra en el centro de la bola. La figura (b) muestra el layout luego de haber ejecutado la traslación $T_n(\mathbf{p}_i) = \mathbf{p}_i \oplus (-\mathbf{n})$, la cual ubica el nodo **n** en el centro de la bola. La figura (c) muestra el *layout* con el nodo **m** en el centro de la bola. Este último posicionamiento resulta de aplicar a (b) la concatenación de la traslación inversa $\mathbf{p}_i = T_n(\mathbf{p}_i) \boxplus (-\mathbf{r})$ con la traslación $T_m(\mathbf{p}_i) = \mathbf{p}_i \oplus (-\mathbf{m})$. La primera transformación vuelve al posicionamiento original del árbol (figura (a)) y la última ubica el nodo **m** en el centro de la bola. (Página 84)

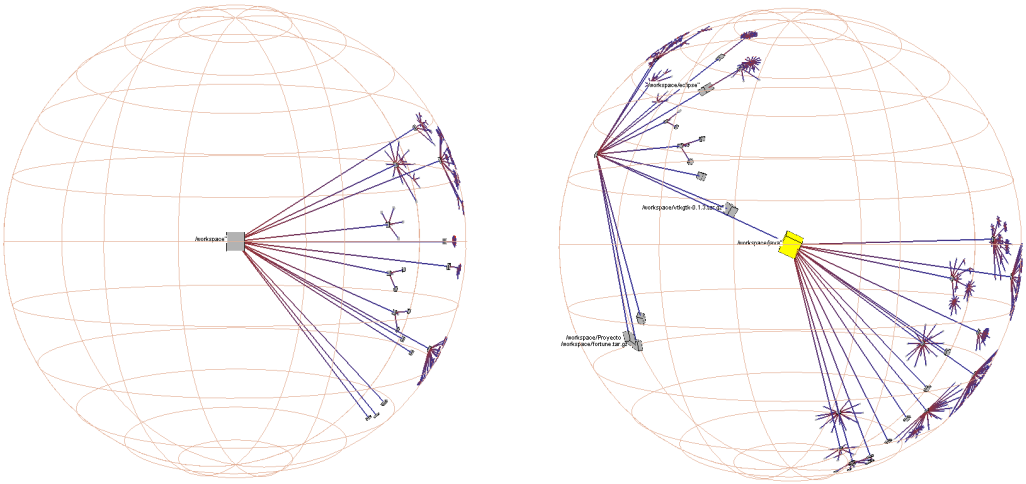


(a) Porción del árbol con máximo detalle.

(b) *Tooltip* sobre un nodo.

(c) *Tooltip* sobre un cono.

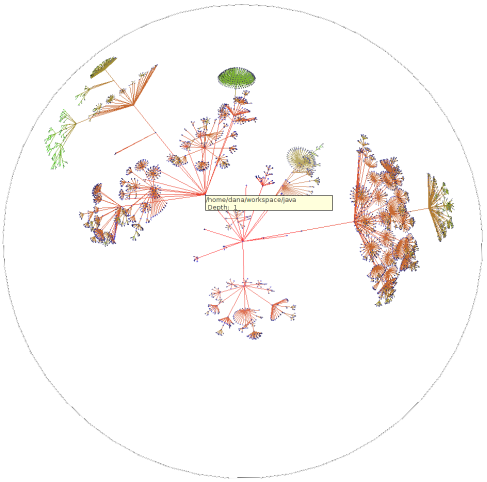
Figura 6.9: Ejemplo del zoom semántico sobre nodos y conos usando *tooltips* para mostrar información adicional. (Página 84)



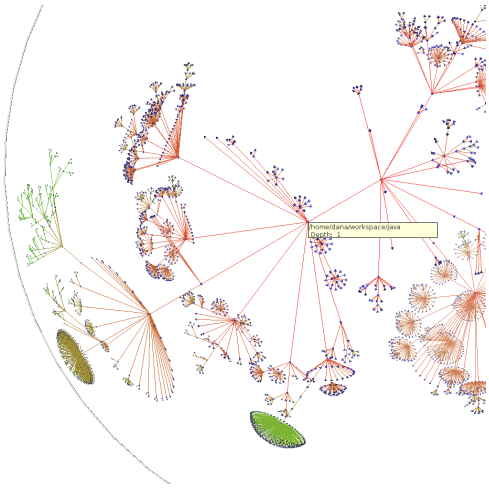
(a) Directorio `workspace` visualizado con H3. **(b)** El mismo directorio. El subdirectorio `workspace/java` está ubicado en el centro de la esfera.

Figura 6.10: Cómo obtener más detalle visualizando con H3. (Página 85)

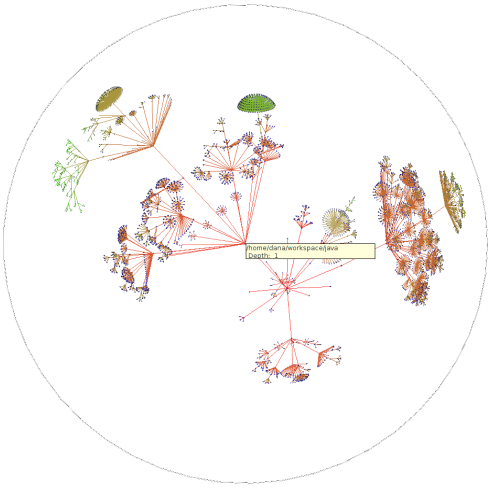
(a) Directorio `workspace` visualizado con Gyrolayout.



(b) Más detalles del subdirectorio `workspace/java` a través de zoom semántico.

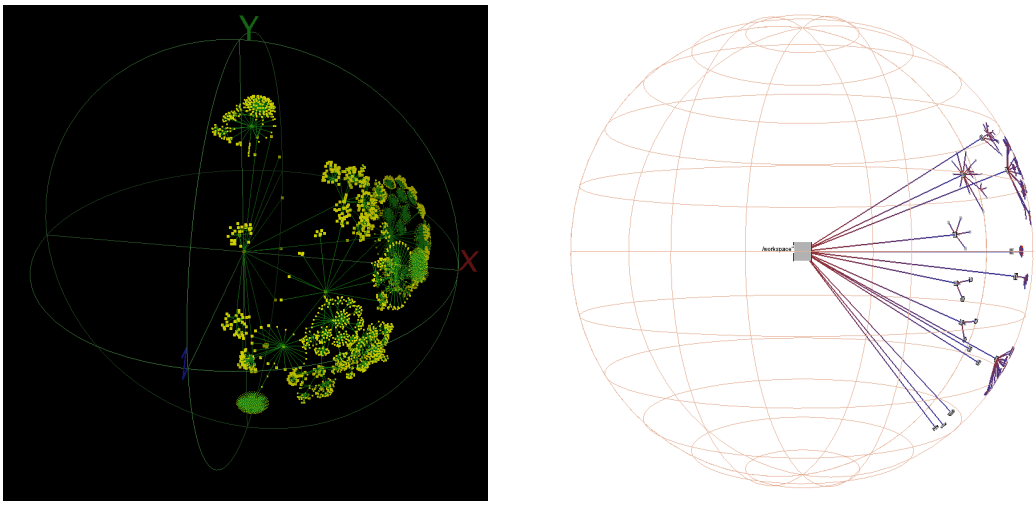


(c) Más detalles del subdirectorio `workspace/java` a través de zoom, paneo y rotación.



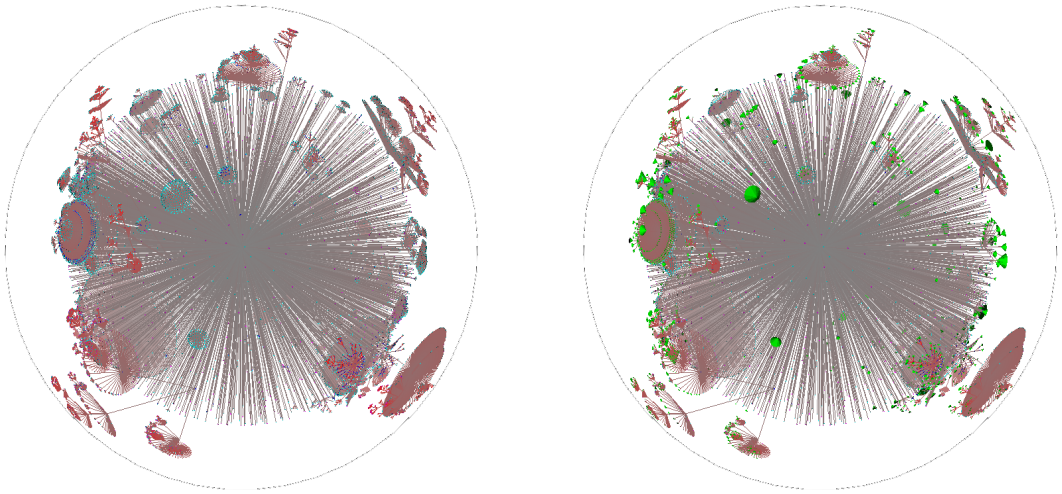
(d) Más detalles del subdirectorio `workspace/java` moviendo el correspondiente nodo al centro de la esfera.

Figura 6.11: Cómo obtener más detalles visualizando con Gyrolayout. El directorio `workspace` tiene 4994 nodos y altura 11. El color de los arcos varía entre rojo, para los arcos con origen en la raíz del árbol, y verde, para los arcos con destino en un nodo de máxima profundidad. (Página 85)



(a) Walrus **(b)** H3

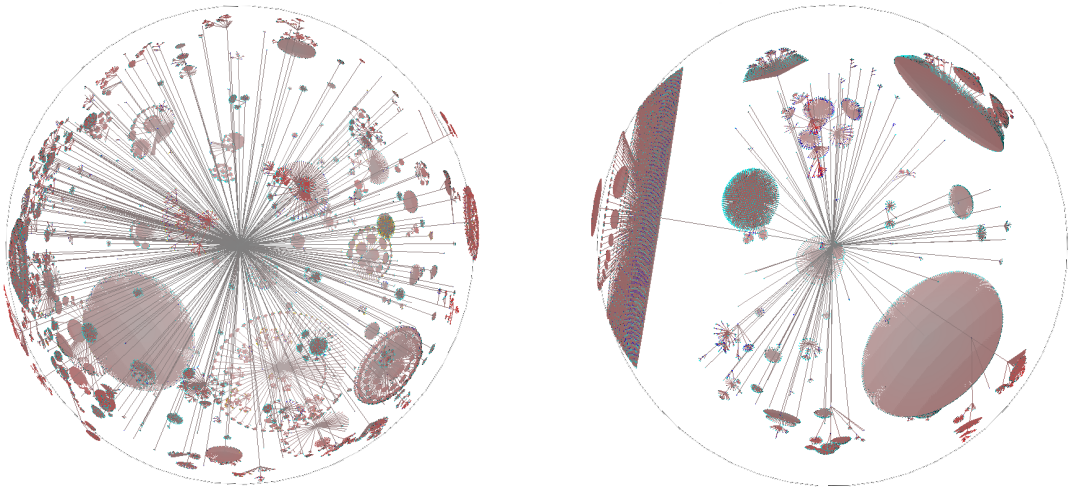
Figura 6.12: El mismo conjunto de datos de la figura 6.11 (un directorio con 4994 nodos y altura 11) visualizado con las otras dos aplicaciones comparadas. Tanto H3 como Walrus ubican los hijos de la raíz de forma que ocupen sólo la superficie de media esfera. Gyrolayout los ubica ocupando toda la superficie de la esfera. (ver figura 6.11(a)). (Página 85)



(a) El directorio `/usr/lib` visualizado con el máximo detalle.

(b) El directorio `/usr/lib` con los directorio que contienen solo archivos colapsados.

Figura 6.13: El direcotiro `/usr/lib`, el cual tiene 25 253 archivos y 2600 subdirectorios, con dos niveles de detalle diferentes. El primer nivel de nodos tiene 2154 archivos and 196 directorios, y el árbol tiene altura 13. En la figura (b) se utiliza un color diferente para los conos y resaltar la idea de subárboles colapsados. (Página 86)



(a) El directorio `/usr/share`, que tienen 127 729 archivos, 10 814 subdirectorios y altura 13.

(b) El direcotrio `windows`, el cual tiene 68 539 archivos, 15 305 subdirectorios y altura 11.

Figura 6.14: Diferentes árboles con diferentes cantidades de nodos internos y hojas. La Figura (a) muestra el directorio `/usr/share` de un sistema Linux, mientras que la Figura (b) muestra el directorio `windows` de un sistema Windows Vista. (Página 86)

Capítulo 7

Predicción de la visibilidad en la visualización de árboles

Se presentan dos métricas, una asociada a los treemaps y la otra asociada al Gyrolayout. Estas métricas permiten predecir la visibilidad de los nodos en la visualización resultante.

7.1. Introducción

Al momento de visualizar grandes árboles hay que tener en cuenta si será suficiente con una técnica básica de visualización o hará falta utilizar técnicas complementarias (como, por ejemplo, *clusterización*). En lo que respecta a la gran cantidad de nodos y a la escalabilidad visual de la técnica elegida, es importante preguntarse:

¿será posible representar todos los nodos, aunque sea con un único píxel?

Dado que es necesario contar con métricas propias de las visualizaciones de árboles, tanto para la representación implícita como para la explícita, se eligieron los *treemaps* como una técnica representativa de la visualización implícita de árboles, y por otro lado, el Gyrolayout (una implementación novedosa de un diagramado hiperbólico de árboles) como la técnica representativa de la visualización explícita de árboles.

En ambos casos y teniendo en cuenta que se busca contar con elementos que, dado un determinado conjunto de datos, permitan seleccionar una técnica de visualización adecuada, es necesario contar con medidas que predigan la calidad de la visualización. En este capítulo se detallan las medidas que fueron desarrolladas para los árboles mencionados y que dado un conjunto de datos y, de ser necesario, las características propias de la visualización permitan predecir la calidad de la visualización resultante en función de la escalabilidad visual.

7.2. Predicción de la visibilidad en *treemaps*

Si bien se han desarrollado algoritmos que mejoran la calidad de la subdivisión del *treemap*, y por lo tanto mejoran la calidad de la visualización, no se han definido medidas que permitan *predecir* la calidad de esta visualización.

En esta sección se define una métrica particular de los *treemaps* que, en función de los parámetros de la visualización y de los datos, cuantifica la visibilidad de los nodos del árbol en el *treemap* sin necesidad de *renderizar* la visualización.

7.2.1. Una medida para cuantificar la visibilidad

En esta sección se presenta una medida que busca cuantificar la cantidad de nodos que serán visibles en un *treemap* con subdivisión *Slice and Dice*. Para desarrollar la medida se considerará que únicamente interesa visualizar la estructura del árbol y por lo tanto, no se consideran hojas o nodos internos con diferentes pesos o tamaños, sino que se asume que todos las hojas son de igual importancia, y por lo tanto de igual peso, y el peso de los nodos internos depende de la cantidad de hojas que contengan.

Considerando ciertas restricciones que imponga el usuario, como tamaño de la visualización, o la separación entre nodos, llamaremos *porcentaje de nodos visibles* a la relación entre aquellos nodos que se representan con al menos un píxel y el total de nodos del árbol.

Para calcular el *porcentaje de nodos visibles* es necesario contar con las dimensiones del mínimo rectángulo necesario para visualizar cada uno de los nodos del árbol. En particular, las dimensiones mínimas necesarias para visualizar la raíz del árbol, son las necesarias para visualizar el árbol en su totalidad. Para obtener este dato, se calcula la máxima subdivisión horizontal y vertical de cada uno de los nodos del árbol.

En esta sección se presentarán los diferentes cálculos intermedios necesarios para calcular la medida final.

Máxima subdivisión horizontal y vertical

La máxima subdivisión horizontal y vertical de un árbol es la máxima subdivisión horizontal y vertical de su nodo raíz (para simplificar la lectura, también se lo llamará simplemente *máxima subdivisión*, sin especificar que es horizontal y vertical). Luego, la máxima subdivisión de un nodo se calcula recursivamente:

- Si el nodo n es una hoja, la máxima subdivisión tanto horizontal como vertical es 1.
- Si el nodo n es un nodo interior, el cálculo dependerá de si sus hijos se disponen de forma horizontal o vertical.
    - Si los hijos del nodo n se organizan de forma horizontal (subdivisión horizontal) la máxima subdivisión horizontal de n será la suma de las subdivisiones

horizontales de sus hijos y la máxima subdivisión vertical será el máximo entre las subdivisiones verticales de sus hijos.

- Análogamente, si los hijos del nodo n se organizan de forma vertical (subdivisión vertical), la máxima subdivisión horizontal de n será el máximo entre las subdivisiones horizontales de sus hijos y la máxima subdivisión vertical será la suma de las subdivisiones horizontales de sus hijos.

Por ejemplo, en el caso que se muestra en la figura 6.1a, la máxima subdivisión horizontal necesaria es 11 y la vertical es 6. El algoritmo 5 especifica cómo se realiza este cálculo.

Algoritmo 5 Máxima subdivisión horizontal y vertical

Entrada: Un nodo n
Sentido de orientación (*horizontal* o *vertical*) de los hijos de n.
Salida: Máxima subdivisión $\mathbf{\Sigma}_n = \left(\sigma_x^n\,,\,\sigma_y^n\right)$ horizontal y vertical de n.

5.1: **si** n es una hoja **entonces**
5.2: $\quad\mathbf{\Sigma}_n \leftarrow (1\,,\,1)$
5.3: **si no**
5.4: $\quad$Sean $m = 1$ y $s = 0$
5.5: $\quad$**para cada** hijo h de n
5.6: $\quad\quad$Sea $\mathbf{\Sigma}_h = \left(\sigma_x^h\,,\,\sigma_y^h\right)$ la máxima subdivisión horizontal y vertical de los descendientes de h, usando la *orientación opuesta* para los hijos.
5.7: $\quad\quad$**si** la orientación de n es vertical **entonces**
5.8: $\quad\quad\quad$Asignar a m el máximo entre m y σ_y^h
5.9: $\quad\quad\quad$Incrementar s en σ_x^h
5.10: $\quad\quad$**si no** ▷ *la orientación de n es horizontal*
5.11: $\quad\quad\quad$Asignar a m el máximo entre m y σ_x^h
5.12: $\quad\quad\quad$Incrementar s en σ_y^h
5.13: $\quad$**si** la orientación de n es vertical **entonces**
5.14: $\quad\quad\mathbf{\Sigma}_n \leftarrow (s\,,\,m)$
5.15: $\quad$**si no** ▷ *la orientación de n es horizontal*
5.16: $\quad\quad\mathbf{\Sigma}_n \leftarrow (m\,,\,s)$

Requerimientos de la visualización

La máxima subdivisión del árbol es una medida inherente a la estructura del árbol específica para la visualización con *treemaps*, que no tiene en cuenta los requerimientos adicionales que puedan existir sobre la visualización. Sin embargo, la medida final debe tener en cuenta dichos requerimientos, en particular, aquí se consideraron los siguientes:

- $\mathbf{V} = (w\,,\,h)$ las dimensiones de la ventana en la cuál se realizará la visualización.
- $\mathbf{S} = (s_x\,,\,s_y)$ la separación entre nodos en la dirección x y en la y.
- $\mathbf{D} = (d_x\,,\,d_y)$ las dimensiones mínimas requeridas para representar una hoja o un nodo interno.

Dimensiones mínimas necesarias

En función de los requerimientos $\mathbf{S}$ y $\mathbf{D}$ se calculan las dimensiones mínimas $\mathbf{M}_n = \left(\mu_x^n\,,\,\mu_y^n\right)$ necesarias para visualizar cada nodo n del árbol:

$$\mathbf{S} = (s_x\,,\,s_y) \quad \mathbf{D} = (d_x\,,\,d_y) \quad \mathbf{\Sigma}_n = \begin{pmatrix} \sigma_x^n & 0 \\ 0 & \sigma_y^n \end{pmatrix}$$

$$\mathbf{M}_n = (\mathbf{S} + \mathbf{D})\,\mathbf{\Sigma}_n + \mathbf{S}$$

$$\mathbf{M}_n = \left(\mu_x^n\,,\,\mu_y^n\right) = \left(\sigma_x^n(s_x + d_x) + s_x\,,\,\sigma_y^n(s_y + d_y) + s_y\right)$$

En particular, si r es la raíz del árbol $\mathcal{A}$, $\mathbf{\Sigma}_r = \mathbf{\Sigma}$ es la máxima subdivisión del árbol y $\mathbf{M}_r = \mathbf{M}$ son las dimensiones mínimas de la ventana necesaria para visualizar el árbol completo.

Porcentaje de nodos visibles

Para representar el árbol $\mathcal{A}$ con los requerimientos $\mathbf{S}$ y $\mathbf{D}$ es necesario contar con una ventana de dimensiones mínimas $\mathbf{M}$; sin embargo, se dispone (como parte de los requerimientos) de una ventana de dimensiones $\mathbf{V}$. Luego, al visualizar el árbol en dicha ventana, a un nodo n cualquiera que requiera una subventana de dimensiones mínimas $\mathbf{M}_n$ le corresponderá una subventana de dimensiones $\mathbf{V}_n$, donde:

$$\mathbf{M} = \begin{pmatrix} \mu_x & 0 \\ 0 & \mu_y \end{pmatrix} \quad \mathbf{V} = \begin{pmatrix} w & 0 \\ 0 & h \end{pmatrix} \quad \mathbf{M}_n = \left(\mu_x^n\,,\,\mu_y^n\right)$$

$$\mathbf{V}_n = \mathbf{M}_n\,\mathbf{M}^{-1}\,\mathbf{V}$$

$$\mathbf{V}_n = (w_n\,,\,h_n) = \left(\frac{\mu_x^n w}{\mu_x}\,,\,\frac{\mu_y^n h}{\mu_y}\right)$$

Luego, si $w_n \geq 1$ y $h_n \geq 1$ ($|\mathbf{V}|^2 \geq 2$), se tiene que el nodo n puede ser representado con al menos un píxel en una ventana de dimensiones $\mathbf{V}$. El algoritmo 6 presenta cómo se calcula el porcentaje de nodos visibles que habrá al visualizar un árbol $\mathcal{A}$ con los requerimientos $\mathbf{S}, \mathbf{D}$ y $\mathbf{V}$

7.2.2. Cómo la métrica asiste al usuario en la configuración de un *Treemap*

Esta métrica brinda información que le permite a un sistema semiautomático predecir cuán buena será la visualización de determinado conjunto jerárquico con un *Treemap* básico. En el caso en que la predicción no sea satisfactoria, puede sugerir cambios en los parámetros que mejoren la visualización resultante, es decir, de ser posible, sugerir

Algoritmo 6 Porcentaje de nodos visibles

Entrada: Un árbol $\mathcal{A}$
Dimensiones $\mathbf{V} = (w\,,\,h)$ de la ventana disponible
Ancho y alto mínimos para representar un nodo $\mathbf{D} = (d_x\,,\,d_y)$
Separación entre nodos en ambas direcciones $\mathbf{S} = (s_x\,,\,s_y)$

Salida: Proporción de nodos del árbol $\mathcal{A}$ visibles con las condiciones especificadas

6.1: Sea $\mathbf{M} = (M_x\,,\,M_y)$ la dimensión de la mínima ventana necesaria para visualizar $\mathcal{A}$.

6.2: Sea c la cantidad de nodos i del árbol $\mathcal{A}$ tales que $|\mathbf{V}_i|^2 \geq 2$, donde $\mathbf{V}_i = \mathbf{M}_i\,\mathbf{M}^{-1}\,\mathbf{V}$ y $\mathbf{M}_i = \left(M_x^i\,,\,M_y^i\right)$ es la dimensión de la mínima ventana necesaria para visualizar el nodo i.

6.3: **retornar** $\frac{c}{\text{cantidad de nodos de } \mathcal{A}}$

disminuir la separación entre nodos, disminuir el tamaño mínimo de los mismos, y/o aumentar el tamaño de la ventana.

Si el porcentaje de nodos visibles es bajo y potencialmente mejorable, el sistema semi-automático puede sugerir cambios en los parámetros de la visualización que incrementen el valor de la medida, tales como:

- disminuir la separación entre nodos,
- disminuir el tamaño mínimo deseable de los nodos,
- aumentar el tamaño de la ventana.

Si el porcentaje de nodos visibles no es aceptable, incluso con la mínima separación posible entre nodos ($\mathbf{S} = (1; 1)$), el mínimo tamaño posible de los nodos ($\mathbf{D} = (1; 1)$) y/o con la máxima ventana posible para la visualización ($\mathbf{V}$ = dimensiones del monitor), el sistema debería desestimar el *treemap* básico como una técnica potencialmente aceptable para la visualización del conjunto de datos (ver figuras 3.3 y 3.4).

7.3. Predicción de la visibilidad en el Gyrolayout

Se han definido propiedades *estéticas* de los diagramados de árboles con representación explícita de arcos [Tam13, cap. 5] relacionadas con lograr una percepción atractiva y un diagramado visualmente eficiente. Aunque muchas veces estas propiedades estéticas sean opuestas entre sí, sería deseable preservarlas para lograr una mejor lectura del árbol:

- *Área* Se refiere al área de dibujado del árbol. Los nodos se ubican en posiciones determinadas por una grilla y el área se calcula como la cantidad de puntos contenidos dentro del rectángulo delimitado por la grilla de dibujado. Esta área puede calcularse sólo en aquellos casos en los que los nodos puedan ubicarse en posiciones discretas de la grilla.

- *Relación de aspecto* [Tam13] La relación de aspecto se define como la razón entre el lado más corto y el lado más largo del rectángulo que encierra el árbol. El uso óptimo de la pantalla se logra minimizando el área de la visualización y permitiendo controlar la relación de aspecto.

- *Separación de subárboles* [Cha+96] Sea T_v el subárbol de T con raíz en v. Un diagramado de T cumple la propiedad de separación de subárboles si para dos subárboles de T con nodos disjuntos T_u y T_v, los rectángulos que encierran los diagramados de T_u y T_v no se superponen.

- *Hoja más cercana* [RS08] La hoja más cercana se define como el mínimo entre las distancias euclídeas de la raíz a cada una de las hojas del árbol.

- *Hoja más lejana* [RS08] La hoja más lejana se define como el máximo entre las distancias euclídeas de la raíz a cada una de las hojas del árbol.

- *Tamaño* [Bat+99] El tamaño del diagramado es el mayor lado del rectángulo más pequeño que cubra al árbol.

- *Longitud total de arcos* [Bat+99] La longitud total de arcos es la suma de las longitudes de todos los arcos del diagramado.

- *Longitud promedio de arcos* [Bat+99] La longitud promedio de arcos es el valor promedio de las longitudes de todos los arcos del diagramado.

- *Longitud máxima de arcos* [Bat+99] La longitud máxima de arcos es el máximo entre todas las longitudes de los arcos del diagramado.

- *Longitud uniforme de arcos* [Bat+99] La longitud uniforme de arcos es la varianza de las longitudes de los arcos del diagramado.

- *Resolución angular* [Bat+99] La resolución angular es el mínimo ángulo formado por dos arcos incidentes a un mismo nodo.

- *Simetría* [Bat+99] Representa la identificación visual de simetrías en el diagramado.

Por otro lado, existen propiedades matemáticas definidas sobre grafos y otras sobre árboles en particular; además algunas de las primeras también resultan apropiadas en este último caso. Entre las más relevantes que se han tenido en cuenta en la visualización de árboles se encuentran las que se mencionan a continuación:

- *Altura*

- *Cantidad de nodos, hojas y nodos internos*

- *Entropía* La entropía de un grafo ([NL07; MD12]) se interpreta como la estructura de su información de contenido y puede utilizarse como una medida de complejidad. Esta medida está asociada a una relación de equivalencia definida en un grafo finito. La partición inducida por la relación de equivalencia permite definir una distribución probabilística. Aplicando la fórmula de entropía de Shannon [Sha48] con la distribución se obtiene un valor numérico que sirve como un indicativo de la característica estructural capturada por la relación de equivalencia. En particular, si X representa el invariante de un grafo y τ una relación de equivalencia que particiona X en k subconjuntos de cardinalidad X_i, la entropía $\mathcal{H}(G,\tau)$ se define como sigue:

$$\mathcal{H}(G,\tau) = -\sum_{i=1}^{k} P_i \log(P_i) =$$

  Alta entropía indica que muchos vértices son igualmente importantes, mientras que baja entropía indica que sólo unos pocos vértices son relevantes.

- *Número de Strahler* Inicialmente el número de Strahler fue definido con el objeto de medir el grado de bifurcación de árboles binarios, y luego se extendió [Aub+04] para árboles en general. Sea η el vértice de un árbol, el número de Strahler σ de η se define como

  - Si η es una hoja entonces $\sigma(\eta) = 1$
  - Si η tiene $k+1$ hijos η_i tales que $\sigma(\eta_i) \leq \sigma(\eta_j)$, si $i \leq j$ entonces

$$\sigma(\eta) = \max_{0 \leq i \leq k} \{\sigma(\eta_i) + i\}.$$

  El número de Strahler de un árbol T, $\sigma(T)$, es el número de Strahler asociado a la raíz de T.

Sin embargo, no hay, en la bibliografía, métricas de calidad orientadas a la escalabilidad visual definidas sobre visualizaciones explícitas de árboles, ni tampoco sobre visualizaciones en el espacio hiperbólico (H3, Walrus, HyperbolicBrowser [Lam+95]).

En esta sección se define un parámetro de calidad orientado a la escalabilidad visual para evaluar las visualizaciones obtenidas utilizando el Gyrolayout y además, una métrica sobre los datos a visualizar que permite predecir el parámetro de calidad definido anteriormente.

7.3.1. Parámetro de calidad: visibilidad de nodos

Se consideró como parámetro para medir la calidad de la visualización la cantidad de nodos visibles, es decir la *visibilidad de los nodos*. Se considerará que un árbol es *completamente visible* si todos sus nodos son visibles y *parcialmente visible* si al menos uno de sus nodos no es visible. Un nodo se considera visible si su representación en la

pantalla necesita al menos un píxel. Con esto se busca establecer si existen nodos que se han acercado tanto a la superficie de la bola (el infinito del espacio hiperbólico) que no serían visibles a pesar de las posibles rotaciones de esta.

Suponiendo una ventana cuadrada de H píxeles de lado, y considerando que se aprovecha al máximo el espacio disponible, se puede estimar la cantidad de píxeles que ocupará cada nodo proyectando ortográficamente la bola de diámetro 2 que representa el espacio hiperbólico sobre tal ventana. Sea α_i el tamaño del nodo i en el espacio hiperbólico, y $A_i = \alpha_i \frac{H}{2}$ la cantidad de píxeles que ocupa el nodo en pantalla; luego, el nodo i se considera visible si $A_i \geq 1$. Un árbol será completamente visible si, para todo nodo i, $A_i \geq 1$. En caso contrario, el árbol será parcialmente visible.

7.3.2. Predicción de árbol completamente visible

En esta sección se propone una métrica que predice si un árbol será completamente visible o no, es decir, permite saber de antemano y *sin aplicar la técnica al conjunto de datos* si un determinado árbol se visualizará completa o parcialmente utilizando la técnica del Gyrolayout.

Para lograr este objetivo se analizaron diferentes propiedades de los árboles y se compararon con la visibilidad de árbol evaluada en una ventana de 1000×1000 píxeles. Para este análisis se estudiaron 144 árboles distintos, entre los que se encontraban varios árboles de directorios reales y otros árboles creados al azar con características particulares (árboles completos de diferentes grados y alturas, árboles con poca cantidad de nodos y mucha altura, y árboles con muchos nodos y poca altura). En particular, la relación *cantidad de nodos–altura* fue encontrada como una de las más prometedoras a la hora de identificar árboles completamente visibles: al graficar el logaritmo de la cantidad de nodos contra el logaritmo de la altura (ver figura 7.1), se puede apreciar que aquellos árboles que se ubican debajo de la línea ℓ son, en general, árboles completamente visibles.

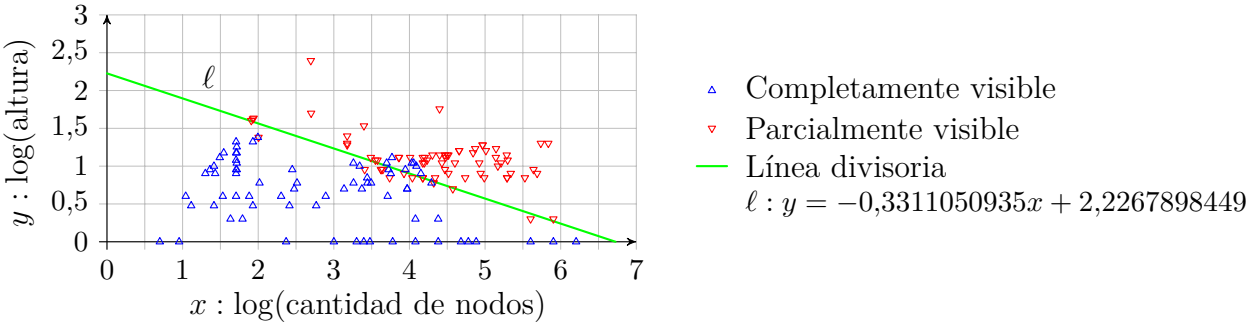


Figura 7.1: Predicción de la visibilidad de nodos en el Gyrolayout. En la figura se muestra cómo la información de cantidad de nodos y altura del árbol se puede utilizar para predecir, con cierto grado de certeza, si todos los nodos serán visibles en una ventana de 1000×1000 píxeles.

En la tabla 7.1 se muestra el resultado de la predicción según los aciertos y los errores en porcentaje sobre el total de las 144 muestras, además de las cantidades absolutas:

		Predicción	
		Completa	Parcial
Visibilidad	Completa	46 % (66)	4 % (6)
	Parcial	8 % (11)	42 % (61)

Tabla 7.1: Sobre un total de 144 muestras, el resultado de la predicción según aciertos y errores en porcentaje y cantidades absolutas.

7.3.3. Asistencia al usuario

Contar con una medida que permita saber de antemano si el árbol de datos que se desea visualizar puede ser completamente visible presenta una buena guía a la hora de elegir la técnica a utilizar para visualizar el árbol. La métrica propuesta es un primer avance en la definición de una métrica que determine cuán visible será un árbol visualizado con Gyrolayout.

A partir de la métrica propuesta, un sistema semiautomático puede tener en cuenta que:

- si el árbol se ubica por debajo de la línea divisoria, será completamente visible.
- si el árbol se ubica por encima de la línea divisoria, será parcialmente visible.

Sin embargo, si el árbol se ubica en la cercanía de la línea divisoria, no es posible saber si será completa o parcialmente visible.

Igualmente, y a pesar de la incertidumbre existente, es posible que para una gran cantidad de árboles, la métrica ayude al sistema (y por lo tanto al usuario) a predecir correctamente el desempeño del Gyrolayout a la hora de visualizar determinado árbol de datos.

7.4. Conclusiones

En este capítulo se definió una métrica particular para *treemaps*, y se presentó una propuesta para el Gyrolayout. Ambas métricas, más allá de las diferencias, buscan predecir la cantidad de nodos visibles para un árbol particular utilizando cada una de las técnicas. En el caso de los *treemaps* la medida devuelve un valor numérico que indica qué porcentaje de nodos será representado con al menos un píxel; en cambio, en el caso del Gyrolayout, la medida devuelve un valor binario, que indica si el árbol será *completamente visible* o *parcialmente visible* en una ventana de 1000×1000 píxeles. Esta medida es aplicable tanto al árbol completo como al árbol colapsado.

Capítulo 8

Conclusiones y Trabajo Futuro

Se presentan las conclusiones finales y el trabajo futuro que puede desarrollarse a partir del realizado en esta tesis.

8.1. Conclusiones

Con el objetivo de contribuir en la toma de decisiones a lo largo del pipeline de visualización, en esta tesis se ha propuesto la aplicación de métricas específicas de escalabilidad visual para técnicas de visualización representativas con el propósito de *predecir* cómo estas se desempeñarán en función de sus factores limitantes para un determinado conjunto de datos.

A partir de la división de los factores limitantes de las técnicas en factores externos e internos, se propuso la definición de métricas orientadas a la escalabilidad visual que cuantifiquen los factores limitantes internos de estas. Dado que, en general, un gran factor limitante de las técnicas de visualización es la oclusión o la superposición debido al reducido espacio disponible para realizar las visualizaciones, se definieron medidas que permiten predecir la *visibilidad* de los datos en la visualización. En particular, se propusieron métricas para predecir la visibilidad en los *scatterplots*, en los *Treemaps* y en el Gyrolayout y se propuso la integración de estas métricas al proceso de visualización. Adicionalmente, se desarrolló un *layout* particular para la visualización de árboles.

Específicamente podemos agrupar las contribuciones en:

- *Gyrolayout.* Se desarrolló una implementación novedosa de un *layout* hiperbólico 3D con capacidades multirresolución. Si bien inicialmente se representan todos los nodos, cuando la cantidad de estos es grande y ofuscan la vista es posible colapsar subárboles para reducir la cantidad de datos visibles tratando de conservar la información. Como parte del desarrollo del Gyrolayout se propuso una extensión del Teselado baricéntrico pesado de Voronoi para subdividir la superficie de una esfera.

- *Escalabilidad visual en scatterplots y árboles.* Se desarrollaron métricas específicas de escalabilidad visual tanto para *scatterplots* como para árboles.

En el caso de los *scatterplots* se definió una fórmula que modela la visibilidad de datos distribuidos según una distribución normal en un *scatterplot* en función de la cantidad de datos, el tamaño de la ventana y el tamaño del glifo; se considera tanto la ventana como el glifo cuadrados.

En lo referido a los árboles se consideraron dos casos, los *treemaps* y el Gyrolayout. En el caso de los *treemaps* se presentó un algoritmo que calcula el porcentaje de nodos visibles según las propiedades definidas para la visualización (dimensiones mínimas de los nodos, separación entre nodos y dimensiones de la ventana). Para el Gyrolayout se propone una propiedad del árbol como métrica para predecir la visibilidad de sus nodos en la visualización, permitiendo distinguir entre aquellos árboles que serán totalmente visibles de aquellos parcialmente visibles.

- *Integración de medidas de escalabilidad visual con el Modelo Unificado de Visualización.* Se integró la propuesta de utilización de métricas al MUV, permitiendo asistir al usuario en la etapa de elección de la técnica más apropiada para un dado conjunto de datos.

  La definición de medidas que predigan algún factor importante de las técnicas de visualización en función del conjunto de datos y de las características particulares de su instanciación, permite predecir aspectos del resultado final de la técnica y por lo tanto distinguir entre aquellas potencialmente efectivas y aquellas que no lo son. Además, estas le permiten al usuario encontrar una configuración aceptable de la técnica sin necesidad de concretar la visualización, incluyendo las capacidades multirresolución de las técnicas.

  En lo que al desarrollo de nuevas técnicas de visualización se refiere, esta tesis plantea la necesidad de que estas estén acompañadas de un análisis de sus factores limitantes internos y de métricas propias que permitan predecir su desempeño y estimar los parámetros más apropiados para visualizar un dado conjunto de datos.

8.2. Trabajo Futuro

En esta tesis se plantea la necesidad de que cada técnica de visualización tengan asociado un conjunto de métricas que indiquen cómo se desempeña la técnica con un determinado conjunto de datos en relación a sus factores limitantes. En base a esto es posible plantear dos tipos de trabajos futuros, aquellos que completan el trabajo aquí presentado y aquéllos derivados que lo extienden:

1. *Trabajos futuros complementarios* Las tres métricas que se presentaron a lo largo de los diferentes capítulos son métricas asociadas a casos particulares de cada técnica, por lo tanto se propone:

   - Extender la métrica definida para los *scatterplots* a otras distribuciones de datos además de la distribución normal, a ventanas no cuadradas y a glifos de diferentes formas.
   - Extender la métrica definida para los *treemaps* a casos en los que haya información asociada a cada nodo y la estructura del árbol no sea lo único importante a visualizar, es decir, que el tamaño de cada nodo sea proporcional a algún dato asociado.
   - Formalizar una métrica para la visibilidad en el Gyrolayout que complete la propuesta presentada, la extienda a diferentes tamaños de ventanas y permita determinar el porcentaje de visibilidad de los nodos.

2. *Trabajos futuros derivados* Dado que el trabajo realizado en esta tesis está enfocado sobre tres técnicas representativas de visualización, para cada una de las cuales se definió una única métrica, se propone:

   - Para las técnicas expuestas en esta tesis, definir nuevas métricas que midan el desempeño de las técnicas en función de otros factores limitantes.
   - Analizar los factores limitantes de otras técnicas de visualización ya existentes y definir métricas para medir el desempeño de otras técnicas en función de los factores limitantes de cada una de ellas.

Apéndice A

Conceptos básicos de geometría hiperbólica

La geometría hiperbólica ([Hes+01]) difiere de la geometría euclídea en la negación del quinto postulado. En particular, en el espacio euclídeo de dos dimensiones (*2D euclídeo* o *plano euclídeo*), dada una recta a y un punto P, $P \notin a$, existe una y sólo una recta b paralela a a que contiene a P. Sin embargo, en el espacio hiperbólico de dos dimensiones (*2D hiperbólico* o *plano hiperbólico*) existen varias rectas diferentes paralelas a a que contienen a P.

Hay varios modelos estándares de geometría hiperbólica: el modelo del hiperboloide, el modelo de Poincaré y el modelo de Klein (también conocido como modelo de Beltrami), entre otros. Tanto el modelo de Klein como el modelo de Poincaré representan todo el espacio hiperbólico en un volumen euclídeo cerrado; además, las rectas en el modelo de Poincaré se corresponden con arcos en el espacio euclídeo, mientras que las rectas en el modelo de Klein son también rectas en el espacio euclídeo. Esta última característica hace al modelo de Klein preferible sobre el modelo de Poincaré.

Los espacios gyrovectoriales [Ung05] son una adaptación de los espacios vectoriales para aplicarse a la geometría hiperbólica; en particular, el álgebra del modelo de Klein es determinada por los espacios gyrovectoriales de Einstein[1]. Dado que los espacios vectoriales son una forma natural de tratar con el espacio Euclídeo y los espacios gyrovectoriales son una adaptación de los espacios vectoriales, se consideró que son una forma natural de tratar con los espacios hiperbólicos. A continuación se presentará una introducción a los espacios gyrovectoriales y sus operaciones.

Sea $\mathcal{V} = (\mathbb{V}, +, \cdot)$ un *espacio vectorial real* con la operación binaria $+$ y el producto interno $\cdot$, y sea $\mathbb{V}_s$ la $s-$esfera de $\mathbb{V}$, para un valor $s > 0$:

$$\mathbb{V}_s = \{\mathbf{v} \in \mathbb{V} : \|\mathbf{v}\| < s\}.$$

[1] El espacio gyrovectorial de Möbious determina el álgebra de la geometría hiperbólica del modelo de Poincaré.

La suma de Einstein $\oplus$ (*gyrosuma*) es una operación binaria en $\mathbb{V}_s$ definida como

$$\mathbf{u} \oplus \mathbf{v} = \frac{1}{1 + \frac{\mathbf{u}\cdot\mathbf{v}}{s^2}} \left(\mathbf{u} + \frac{1}{\gamma_\mathbf{u}}\mathbf{v} + \frac{1}{s^2}\frac{\gamma_\mathbf{u}}{1+\gamma_\mathbf{u}}(\mathbf{u}\cdot\mathbf{v})\mathbf{u} \right)$$

donde

$$\gamma_\mathbf{u} = \frac{1}{\sqrt{1 - \frac{\|\mathbf{u}\|^2}{s^2}}}$$

y el producto por un escalar $\otimes$ es definido por

$$\begin{aligned} r \otimes \mathbf{0} &= \mathbf{0} \\ r \otimes \mathbf{v} &= s \tanh\left(r \tanh^{-1} \tfrac{\|\mathbf{v}\|}{s} \right) \tfrac{\mathbf{v}}{\|\mathbf{v}\|} \end{aligned}$$

donde $r \in R$, $\mathbf{v} \in \mathbb{V}_s$ y $\mathbf{v} \neq \mathbf{0}$.

En todos los casos, la norma $\|\cdot\|$, el producto escalar y las operaciones vector-escalar son las que $\mathbb{V}_s$ hereda del espacio vectorial $\mathcal{V}$. Notar que la gyrosuma no es conmutativa ni asociativa.

El espacio gyrovectorial de Einstein $(R^3_s, \oplus, \otimes)$ es equivalente al modelo de Beltrami para la geometría hiperbólica, la cual es representada por una esfera de radio s dentro del espacio euclídeo R^3. En nuestro *layout* asumimos $s = 1$, sin embargo, en lo que sigue de esta sección se mantendrá s como una posible variable, para preservar la generalidad de las fórmulas.

En este punto, es necesario definir algunos conceptos importantes para poder usar las operaciones del espacio gyrovectorial de Einstein para implementar la geometría hiperbólica de la visualización de árboles que proponemos.

Gyropunto y gyroposición

Llamaremos *gyropunto* a los elementos del espacio gyrovectorial para evitar ambigüedad entre estos y los puntos del espacio euclídeo. Una *gyroposición* es una posición en el espacio determinada por un gyropunto.

Gyrodiferencia

La gyrodiferencia $\ominus$ se define como

$$\mathbf{a} \ominus \mathbf{b} = \mathbf{a} \oplus -\mathbf{b} = \mathbf{a} \oplus (-1 \otimes \mathbf{b}).$$

Gyrométrica y métrica

La métrica del espacio gyrovectorial de Einstein se define como

$$h(\mathbf{a}, \mathbf{b}) = \tanh^{-1} \frac{d(\mathbf{a}, \mathbf{b})}{s} = \phi_{\mathbf{b}\ominus\mathbf{a}}$$

donde $d(\mathbf{a},\mathbf{b}) = \| \ominus \mathbf{a} \oplus \mathbf{b}\| = \|\mathbf{b} \ominus \mathbf{a}\|$ es una gyrométrica del espacio gyrovectorial de Einstein, y $\phi_{\mathbf{v}} = \tanh^{-1} \frac{\|\mathbf{v}\|}{s}$.

Gyrovector normalizado

Se considera que un gyrovector $\mathbf{v}$ está *normalizado* si la métrica $h(\mathbf{0},\mathbf{v}) = \phi_{\mathbf{v}\ominus\mathbf{0}} = \phi_{\mathbf{v}} = 1$. Por consiguiente, $\|\mathbf{v}\| = s \tanh 1$. Dado un gyrovector $\mathbf{u}$, el gyrovector normalizado de $\mathbf{u}$ será

$$\frac{1}{\phi_{\mathbf{u}}} \otimes \mathbf{u} = \frac{s \tanh 1}{\|\mathbf{u}\|}\mathbf{u}.$$

Gyrodirecciones y ángulos

Una gyrodirección es un gyrovector $\mathbf{v}$ tal que $\|\mathbf{v}\| = 1$. El ángulo entre dos gyrovectors es el ángulo entre las dos gyrodirecciones asociadas. Sean $\mathbf{u}$ y $\mathbf{v}$ dos gyrovectores, y α el ángulo entre ellos, entonces $\cos\alpha = \frac{\mathbf{u}}{\|\mathbf{u}\|} \cdot \frac{\mathbf{v}}{\|\mathbf{v}\|}$.

Base Ortonormal

Una base ortonormal es un conjunto de 3 gyrovectores normalizados $\langle \mathbf{b}_1, \mathbf{b}_2, \mathbf{b}_3 \rangle$ dónde el ángulo entre cada par de ellos es $\pi/2$:

$$\|\mathbf{b}_1\| = \|\mathbf{b}_2\| = \|\mathbf{b}_3\| = s \tanh 1$$

$$\mathbf{b}_1 \cdot \mathbf{b}_2 = \mathbf{b}_1 \cdot \mathbf{b}_3 = \mathbf{b}_2 \cdot \mathbf{b}_3 = \cos \pi/2 = 0$$

Gyrovector dirigido

Un gyrovector dirigido $\mathbf{pq}$, determinado por la cola $\mathbf{p}$ y la cabeza $\mathbf{q}$, está definido por la diferencia

$$\mathbf{pq} = \ominus\mathbf{p} \oplus \mathbf{q}.$$

Translaciones y rotaciones

Una traslación T de un gyropunto $\mathbf{p}$ por un gyrovector $\mathbf{t}$ se define por la ecuación $\mathbf{p}' = T_{\mathbf{t}}(\mathbf{p}) = \mathbf{p} \oplus \mathbf{t}$. La traslación inversa T^{-1} de una traslación T es $\mathbf{p} = T_{\mathbf{t}}^{-1}(\mathbf{p}') = \mathbf{p}' \boxplus -\mathbf{t}$, donde $\boxplus$ se denomina *cosuma* y se define por la ecuación

$$\mathbf{u} \boxplus \mathbf{v} = 2 \otimes \frac{\gamma_{\mathbf{u}}\mathbf{u} + \gamma_{\mathbf{v}}\mathbf{v}}{\gamma_{\mathbf{u}} + \gamma_{\mathbf{v}}}.$$

Sea l una recta definida por dos gyropuntos diferentes $\mathbf{p}$ y $\mathbf{q}$. Una rotación R de un gyropunto $\mathbf{v}$ alrededor de la recta l por un ángulo θ es definida en [PG92] por la ecuación

$$R_{\mathbf{l},\theta}(\mathbf{v}) = T_{\mathbf{t}}^{-1} \left(R_{\mathbf{u},\theta}^{euc} (T_{\mathbf{t}}(\mathbf{v})) \right),$$

dónde

$$\mathbf{t} = -\frac{\mathbf{p}\cdot(\mathbf{p}-\mathbf{q})}{(\mathbf{p}-\mathbf{q})\cdot(\mathbf{p}-\mathbf{q})}\mathbf{q} - \frac{\mathbf{q}\cdot(\mathbf{q}-\mathbf{p})}{(\mathbf{q}-\mathbf{p})\cdot(\mathbf{q}-\mathbf{p})}\mathbf{p},$$

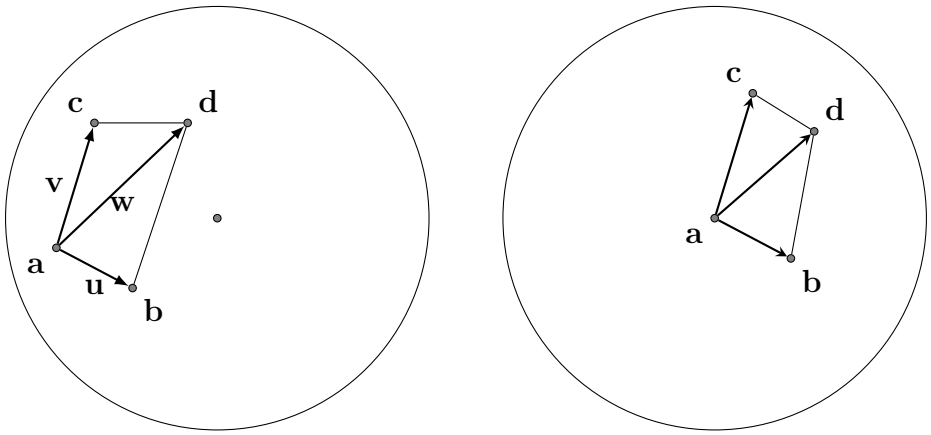
$$\mathbf{u} = \frac{\mathbf{p}-\mathbf{q}}{\|\mathbf{p}-\mathbf{q}\|}$$

y $R^{euc}_{\mathbf{u},\theta}$ es la rotación habitual del espacio euclídeo de tres dimensiones.

Gyroparalelogramo

Sean **a**, **b** y **c** tres gyropuntos. Luego, los cuatro puntos **a**, **b**, **c** y **d** son los vértices del gyroparalelogramo **abdc**, si $\mathbf{d} = (\mathbf{b} \boxplus \mathbf{c}) \ominus \mathbf{a}$.

Ley de Einstein de la suma del gyroparalelogramo Sea **abcd** un gyroparalelogramo, luego $(\ominus\mathbf{a} \oplus \mathbf{b}) \boxplus (\ominus\mathbf{a} \oplus \mathbf{c}) = (\ominus\mathbf{a} \oplus \mathbf{d})$ (ver figura A.1).



(a) Sea **d** un gyropunto tal que $\mathbf{d} = (\mathbf{b} \boxplus \mathbf{c}) \oplus \mathbf{a}$, entonces **abcd** es un gyroparalelogramo. Además, sean $\mathbf{u} = \ominus\mathbf{a} \oplus \mathbf{b}$, $\mathbf{v} = \ominus\mathbf{a} \oplus \mathbf{c}$ y $\mathbf{w} = \ominus\mathbf{a} \oplus \mathbf{d}$, luego $\mathbf{u} \boxplus \mathbf{v} = \mathbf{w}$.

(b) En el caso particular que $\mathbf{a} = \mathbf{0}$, la ley de la suma del gyroparalelogramo se reduce a $\mathbf{d} = \mathbf{b} \boxplus \mathbf{c}$.

Figura A.1: Ley de Einstein de la suma del gyroparalelogramo. La figura (a) ilustra el caso general de la ley de la suma del gyroparalelogramo. Por otro lado, la figura (b) muestra el caso particular en que uno de los vértices del gyroparalelogramo es coincidente con el origen.

Apéndice B

Extensiones del Teselado de Voronoi

Se presenta el Teselado esférico, baricéntrico y pesado de Voronoi aplicado tanto al Layout Esférico 3D en Larrea et al. [Lar+09] como aplicado al Gyrolayout en Urribarri et al. [Urr+13]. Para esto, es necesario definir previamente los conceptos de Teselado de Delaunay, Teselado de Voronoi, Teselado esférico de Voronoi y Teselado esférico pesado de Voronoi.

B.1. Teselado de Delaunay

Dado un conjunto de puntos $P = \{p_1, p_2, \ldots, p_n\}$ en R^m, un Teselado de Delaunay es un conjunto de k símplex $t_1, t_2, \ldots t_k$ con vértices en P tales que ningún punto de P está en el interior de la hiper-esfera circunscrita a ningún símplex t_i.

Una Triangulación de Delaunay (Teselado de Delaunay bidimensional) es un conjunto de triángulos con vértices en P tales que ningún punto de P está en el interior de la circunferencia circunscrita a ningún triángulo de la triangulación.

B.2. Teselado de Voronoi

Dado un conjunto de puntos (generadores) $P = \{p_1, p_2, \ldots, p_n\}$ en R^m, un Teselado de Voronoi es un conjunto de n regiones $V(p_i)$, donde un punto $q \in R^m$ está dentro de la región $V(p_i)$ si y sólo si distancia$(p_i, q) <$ distancia(p_j, q) para todo $p_i, p_j \in P, i \neq j$.

El Teselado de Voronoi y el Teselado de Delaunay son duales: toda región de Teselado de Voronoi (es decir, todo generador) se corresponde con un vértice del Teselado de Delaunay y por cada par de regiones adyacentes en el Teselado de Voronoi hay una arista entre los correspondientes vértices del Teselado de Delaunay.

B.3. Teselado esférico de Voronoi

Un Teselado esférico de Voronoi es un Teselado de Voronoi de la superficie de la esfera. En este caso, el conjunto P es un conjunto de puntos pertenecientes a la superficie

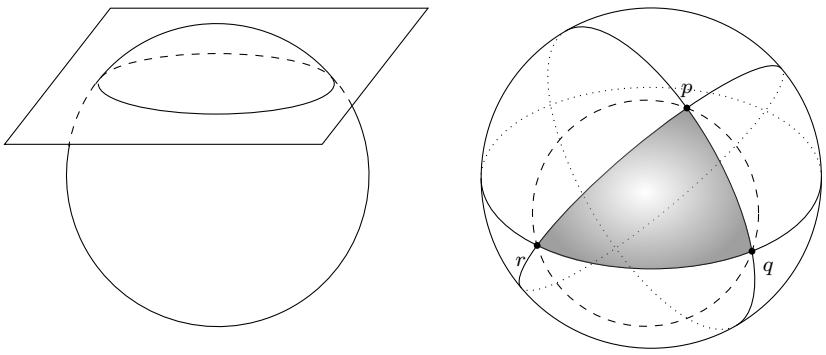
$\mathcal{S} = \{(x, y, z) \in R^3 : x^2 + y^2 + z^2 = 1\}$, y las regiones $V(p_i)$ son los puntos $q \in \mathcal{S}$ que satisfacen distancia$(p_i, q) <$ distancia(p_j, q) para todo $p_i, p_j \in P, i \neq j$.

B.4. Teselado esférico pesado de Voronoi

Un Teselado esférico pesado de Voronoi es un Teselado esférico de Voronoi en el que cada generador p_i tiene asociado un peso w_i, y la distancia entre un punto q y un generador p_i es la distancia pesada w-distancia$(a, w_a, x) = |a - x|^2 - w_a$, donde $|.|$ es la distancia euclídea.

La idea general del algoritmo que calcula el Teselado esférico pesado de Voronoi de los puntos P sobre la esfera está esquematizado en el algoritmo 7. Dado un conjunto de puntos P sobre la superficie de la esfera y el conjunto de pesos W con los pesos correspondientes a cada punto, el algoritmo calcula el Teselado esférico pesado de Voronoi de la superficie de la esfera en función de P y W.

La Triangulación de Delaunay de los puntos sobre la superficie de la esfera se calcula como el *convex hull* de esos puntos, basado en que si $\triangle pqr$ es una cara del *convex hull*, entonces existe un plano P que contiene a los puntos p, q y r, el cual deja un semiespacio vacío a un lado, y todos los puntos restantes del otro lado. La intersección entre P y la esfera S es una circunferencia (ver figura B.1a), que es la circunferencia circunscrita al triángulo esférico $\spadesuit pqr$ (ver figura B.1b). Dentro de la circunferencia circunscrita no hay puntos de P; todos los demás puntos se encuentran fuera de la circunferencia, contenidos en el semiespacio no vacío. Por lo tanto, $\spadesuit pqr$ es un triángulo de Delaunay sobre la superficie de la esfera.



(a) Intersección entre el plano y la esfera

(b) Circunferencia circunscrita al triángulo esférico

Figura B.1: Circunferencia circunscrita al triángulo esférico. (a) Circunferencia resultante de la intersección entre la esfera y un plano. (b) La intersección entre la esfera y el plano es la circunferencia circunscrita del triángulo esférico ya que contiene a los tres vértices del triángulo.

El circuncentro pesado de un triángulo esférico $\spadesuit abc$ donde w_a, w_b y w_c son los pesos de los vértices a, b y c respectivamente, se define como sigue. Sea c el circuncentro del

triángulo $\triangle abc$, c_w el punto coplanar a a, b y c que satisface

$$\text{w-distancia}(a, w_a, x) = \text{w-distancia}(b, w_b, x) = \text{w-distancia}(c, w_c, x),$$

y n la normal del plano que contiene al $\triangle abc$

$$n = \frac{(b-a) \times (c-a)}{|(b-a) \times (c-a)|}.$$

Luego, el circuncentro pesado s_w (ver figura B.2) de $\text{◬}abc$ es la intersección entre la bola unitaria y la semirrecta de dirección n y origen en el punto c'_w, donde c'_w es el punto:

$$c'_w = \frac{(c_w - c)\sqrt{1-(c.c)}}{|c_w - c| + 1} + c$$

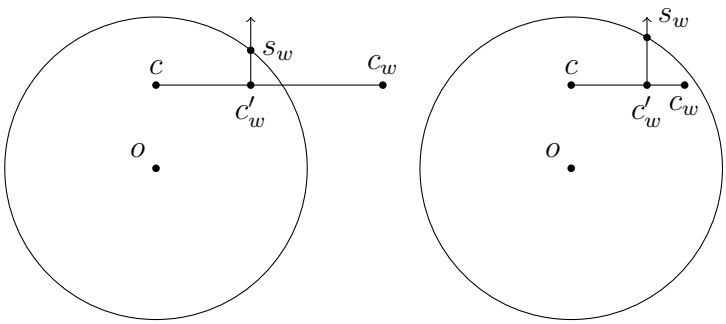


Figura B.2: Corte transversal de la esfera generado por el plano normal $n \times (c_w - c)$, el cual contiene al origen de la esfera. En el corte transversal evidencia cómo h_w se mapea en la bola y el proyectado sobre la superficie para alcanzar el circuncentro pesado.

Algoritmo 7 Teselado esférico pesado de Voronoi (WSVT)

Entrada: Un conjunto de puntos $P = \{p_1, \ldots, p_n\}$.
Un conjunto de pesos $W = \{w_1, \ldots, w_n\}$ donde w_i es el peso de p_i.

Salida: El Teselado esférico pesado de Voronoi $\mathcal{V}$ de la superficie esférica $\mathcal{S}$ en función de P y W.

7.1: Sea $\mathcal{H}$ el *convex hull* de P en R^3. Esto representa la Triangulación de Delaunay de P sobre $\mathcal{S}$. Notar que $\mathcal{H}$ no es pesada.

7.2: Sea $\mathcal{V}$ el Teselado de Voronoi construido como el grafo dual de $\mathcal{H}$: para cada triángulo en $\mathcal{H}$ su circuncentro pesado es un vértice de $\mathcal{V}$. Si dos triángulos en $\mathcal{H}$ son vecinos, entonces sus circuncentros pesados están unidos por una arista en $\mathcal{V}$.

7.3: **retornar** $\mathcal{V}$.

B.5. Teselado esférico, baricéntrico y pesado de Voronoi

Un Teselado baricéntrico de Voronoi (CVT) [Du+99] es un Teselado de Voronoi particular, en el cual el generador de cada región de Voronoi es su centro de masa (baricentro). Un CVT con distancia pesada es apropiado para dividir la superficie en subáreas donde el tamaño de cada una depende del generador en sí mismo y no de su posición en la superficie. Para calcular un Teselado esférico, baricéntrico y pesado de Voronoi (WSCVT) es necesario introducir la definición de baricentro de un triángulo esférico y baricentro de un polígono esférico. Se consideran triángulos y polígonos en la esfera unitaria. El baricentro de un triángulo esférico $\triangle abc$ es $\frac{a+b+c}{|a+b+c|}$. El baricentro de un polígono esférico $v_0, \ldots, v_n$ [Jen08] es

$$\frac{\sum_{i=1}^{n-1} \text{área}(\triangle v_0 v_i v_{i+1}) \;\; \text{baricentro}(\triangle v_0 v_i v_{i+1})}{\sum_{i=1}^{n-1} \text{área}(\triangle v_0 v_i v_{i+1})},$$

donde el área de un triángulo $\triangle abc$ es igual a su exceso esférico E

$$E = 4 \tan^{-1} \sqrt{\tan\left(\frac{s}{2}\right) \tan\left(\frac{s-A}{2}\right) \tan\left(\frac{s-B}{2}\right) \tan\left(\frac{s-C}{2}\right)},$$

donde A, B y C son las longitudes de los lados y s es el semiperímetro.

El algoritmo WSCVT presentado es una adaptación del que computa el Teselado baricéntrico pesado de Voronoi en superficies planas presentado por Balzer y Deussen [BD05b]; el algoritmo original subdivide una superficie plana finita; este algoritmo se extendió a 3D para subdividir la superficie de una esfera. La idea general del algoritmo es construir el WSVT para un conjunto de generadores y luego reemplazar cada generador con el baricentro de su correspondiente región de Voronoi, hasta alcanzar un error aceptable o un límite preestablecido para la cantidad de iteraciones. Dado que la distancia pesada no es suficiente para controlar el tamaño de cada región, es necesario ajustar el peso de los generadores en cada iteración. Cuando el tamaño de una región es mayor que el tamaño deseado, el peso de su generador debe disminuirse; análogamente, si el tamaño es menor, debe aumentarse el peso del generador.

Es importante remarcar que los valores de los tamaños no son áreas sino porcentajes. Si se asocia el peso de un generador con la superficie de la esfera, el área de una región G_i debe representar el mismo porcentaje sobre toda la superficie de la esfera, que el peso w_i sobre la suma total de pesos. Luego, el valor del tamaño deseado d_i y el valor del tamaño real a_i de una región G_i son

$$d_i = \frac{w_i}{\sum w_i} \qquad a_i = \frac{\text{área}(G_i)}{\text{área}(\mathcal{S})}.$$

El algoritmo se detiene cuando la diferencia entre el valor del tamaño deseado y el valor del tamaño actual de todas las regiones se encuentra por debajo de un dado error ε. El peso ajustado de un generador p_i con peso actual w_i, valor de tamaño deseado d_i y valor

de tamaño real a_i es

$$w_i \left(1 + \frac{a_i - d_i}{d_i}\right) \text{ si el valor es mayor que } \delta\text{; sino, } \delta$$

donde δ es un valor positivo cercano a 0, por ejemplo 10^{-6}, que evita obtener generadores con peso nulo. El algoritmo que genera el WSCVT se esquematiza en el algoritmo 8.

Algoritmo 8 Teselado esférico, baricéntrico y pesado de Voronoi (WSCVT)

▷ *Ubica los nodos en la superficie de la esfera*

Entrada: Un conjunto de pesos $W = \{w_1, \ldots, w_n\}$.
Salida: Un conjunto de puntos $P = \{p_1, \ldots, p_n\}$, cada punto se corresponde con la posición de un nodo, distribuidos según W.

8.1: Sea P una distribución tentativa inicial de puntos sobre una esfera unitaria $\mathcal{S}$. ▷ *Posiblemente una distribución aleatoria*
8.2: Sea D el conjuntos de valores deseados $\left(d_i = \frac{w_i}{\sum w_i}\right)$
8.3: **mientras** $\varepsilon_{\max} > \varepsilon$ ▷ *no se alcanzó un error aceptable*
8.4: Sea $\mathcal{V}$ el WSVT de P.
8.5: Sea $\varepsilon_{\text{real}}$ la diferencia máxima (o promedio) entre los valores deseados (d_i) y los valores reales $\left(\frac{\text{área}(G_i)}{\text{área}(\mathcal{S})}\right)$ de las regiones de $\mathcal{V}$.
8.6: **para cada** región G_i de $\mathcal{V}$
8.7: Sea $p_i \in P$ el generador de la región G_i.
8.8: Reemplazar p_i con el baricentro esférico de G_i.
8.9: **retornar** P.

Bibliografía

[Aub+04] D. Auber, M. Delest, J. P. Domenger, P. Duchon y J. M. Fédou. "New Strahler numbers for rooted plane trees". En: *Third Colloquium on Mathematics and Computer Science*. 2004, págs. 203-215 (véase página 101).

[BC87] Richard A. Becker y William S. Cleveland. "Brushing Scatterplots". En: *Technometrics* 29.2 (1987), págs. 127-142. URL: http://www.jstor.org/stable/1269768 (véase páginas 37, 38).

[BD05a] Michael Balzer y Oliver Deussen. "Voronoi Treemaps". En: *InfoVis'05: Proceedings of the IEEE Symposium on Information Visualization*. Ed. por John Stasko y Matthew O. Ward. Minneapolis, MN, USA: IEEE Computer Society, 2005, págs. 49-56. ISBN: 078039464X. DOI: 10.1109/INFVIS.2005.1532128 (véase página 74).

[BD05b] Michael Balzer y Oliver Deussen. "Voronoi Treemaps". En: *IEEE Symposium on Information Visualization (InfoVis 2005)*. 2005, págs. 49-56 (véase página 116).

[BL07] Renaud Blanch y Éric Lecolinet. "Browsing Zoomable Treemaps: Structure-Aware Multi-Scale Navigation Techniques". En: *IEEE Transactions on Visualization and Computer Graphics* 13.6 (2007), págs. 1248-1253. DOI: 10.1109/TVCG.2007.70540 (véase página 75).

[BM12] Richard Brath y Peter MacMurchy. "Sphere-based Information Visualization: Challenges and Benefits." En: *IV*. Ed. por Ebad Banissi, Stefan Bertschi, Remo Aslak Burkhard, Urska Cvek, Martin J. Eppler, Camilla Forsell, Georges G. Grinstein, Jimmy Johansson, Sarah Kenderdine, Francis T. Marchese, Carsten Maple, Marjan Trutschl, Muhammad Sarfraz, Liz J. Stuart, Anna Ursyn y Theodor G. Wyeld. IEEE Computer Society, 2012, págs. 1-6. ISBN: 978-1-4577-0868-8. URL: http://dblp.uni-trier.de/db/conf/iv/iv2012.html#BrathM12 (véase página 76).

[BS04] Enrico Bertini y Giuseppe Santucci. "Quality Metrics for 2D Scatterplot Graphics: Automatically Reducing Visual Clutter". En: *Proc. of Symposium on Smart Graphics*. Vol. 3031. 2004, págs. 77-89 (véase páginas 57, 60).

[BS06a] Enrico Bertini y Giuseppe Santucci. "Give chance a chance- modeling density to enhance scatter plot quality through random data sampling". En: *Information Visualization* 5.2 (2006), págs. 95-110 (véase páginas 57, 60).

[BS06b] Enrico Bertini y Giuseppe Santucci. "Visual quality metrics". En: *Proceedings of the 2006 AVI workshop on Beyond time and errors: novel evaluation methods for information visualization*. BELIV '06. Venice, Italy: ACM, 2006, págs. 1-5. ISBN: 1-59593-562-2. DOI: 10.1145/1168149.1168159. URL: http://doi.acm.org/10.1145/1168149.1168159 (véase páginas 26-28).

[Bat+99] Giuseppe Di Battista, Peter Eades, Roberto Tamassia y Ioannis G. Tollis. *Graph Drawing: Algorithms for the Visualization of Graphs*. 1st. Upper Saddle River, NJ, USA: Prentice Hall PTR, 1999. ISBN: 0133016153 (véase página 100).

[Bec97] Barry G. Becker. *Volume Rendering for Relational Data*. Inf. téc. Silicon Graphics Inc., 1997 (véase páginas 37, 39).

[Bed+02] Benjamin B. Bederson, Ben Shneiderman y Martin Wattenberg. "Ordered and quantum treemaps: Making effective use of 2D space to display hierarchies". En: *ACM Transactions on Graphics* 21.4 (2002), págs. 833-854. DOI: 10.1145/571647.571649 (véase página 74).

[Ber+11] Enrico Bertini, Andrada Tatu y Daniel Keim. "Quality Metrics in High-Dimensional Data Visualization: An Overview and Systematization". En: *IEEE Transactions on Visualization and Computer Graphics* 17.12 (dic. de 2011), págs. 2203-2212. ISSN: 1077-2626. DOI: 10.1109/TVCG.2011.229. URL: http://dx.doi.org/10.1109/TVCG.2011.229 (véase páginas 26, 27, 60).

[Bon12] Roberto Bonola. *Non Euclidean-Geometry. A critical and historical study of its development*. Chicago. The Open Court Publishing Company, 1912 (véase página 78).

[Bor+13] Rita Borgo, Johannes Kehrer, David H.S. Chung, Eamonn Maguire, Robert S. Laramee, Helwig Hauser, Matthew Ward y Min Chen. "Glyph-based Visualization: Foundations, Design Guidelines, Techniques and Applications". En: *Eurographics State of the Art Reports*. EG STARs. http://diglib.eg.org/EG/DL/conf/EG2013/stars/039-063.pdf. Girona, Spain: Eurographics Association, mayo de 2013, págs. 39-63. URL: http://www.cg.tuwien.ac.at/research/publications/2013/borgo-2013-gly/ (véase páginas 12, 36).

[Bra97] Richard Brath. "Metrics for effective information visualization". En: *INFOVIS*. IEEE Computer Society, 1997, págs. 108-111. ISBN: 0-8186-8189-6 (véase páginas 56, 60).

[Bru+99] Mark Bruls, Kees Huizing y Jarke van Wijk. "Squarified Treemaps". En: *In Proceedings of the Joint Eurographics and IEEE TCVG Symposium on Visualization*. Press, 1999, págs. 33-42. URL: `http://diglib.eg.org/EG/DL/WS/VisSym/VisSym00/033-042.pdf` (véase página 74).

[CAI05] CAIDA. *Walrus - Graph Visualization Tool*. 2005. URL: `http://www.caida.org/tools/visualization/walrus/` (véase páginas 76, 85).

[CK95] Jeromy Carriere y Rick Kazman. "Interacting with Huge Hierarchies: Beyond Cone Trees". En: *Proc. IEEE Information Visualization '95*. IEEE Computer Press, 1995, págs. 74-81 (véase página 82).

[CM84] William S. Cleveland y Robert McGill. "The Many Faces of a Scatterplot". En: *Journal of the American Statistical Association* 79.388 (1984), págs. 807-822. URL: `http://www.jstor.org/stable/2288711` (véase páginas 37-39, 46).

[CM97] S. K. Card y J. Mackinlay. "The structure of the information visualization design space". En: *INFOVIS '97: Proceedings of the 1997 IEEE Symposium on Information Visualization (InfoVis '97)*. Washington, DC, USA: IEEE Computer Society, 1997, pág. 92. ISBN: 0-8186-8189-6 (véase página 29).

[CR98] Ed Huai hsin Chi y John Riedl. "An Operator Interaction Framework for Visualization Systems". En: *INFOVIS '98: Proceedings of the 1998 IEEE Symposium on Information Visualization*. North Carolina: IEEE Computer Society, 1998, págs. 63-70. ISBN: 0-8186-9093-3 (véase página 29).

[Car+87] D. B. Carr, R. J. Littlefield, W. L. Nicholson y J. S. Littlefield. "Scatterplot matrix techniques for large N". En: *Journal of the American Statistical Association* 82.398 (1987), págs. 424-436 (véase páginas 10, 18, 37-39).

[Car+99] Stuart K. Card, Jock D. Mackinlay y Ben Shneiderman, eds. *Readings in information visualization: using vision to think*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1999. ISBN: 1-55860-533-9 (véase páginas 1, 83).

[Car03] Stuart Card. "Information Visualization". En: ed. por J. A. Jacko y A. Sears. Lawrence Erlbaum Associates, Inc., 2003. Cap. 26, págs. 509-543 (véase página 19).

[Cha+96] Timothy M. Chan, Michael T. Goodrich, S. Rao Kosaraju y Roberto Tamassia. "Optimizing Area and Aspect Ratio in Straight-Line Orthogonal Tree Drawings". En: *Graph Drawing*. Ed. por Stephen C. North. Vol. 1190. Lecture Notes in Computer Science. Springer, 1996, págs. 63-75. ISBN: 3-540-62495-3 (véase página 100).

[Che73] Herman Chernoff. "The Use of Faces to Represent Points in K-Dimensional Space Graphically". En: *Journal of the American Statistical Association* 68.342 (1973), págs. 361-368 (véase páginas 12, 36, 37).

[Cho+11] Junghong Choi, Oh-hyun Kwon y Kyungwon Lee. "Strata treemaps". En: *ACM SIGGRAPH 2011 Posters*. SIGGRAPH '11. Vancouver, British Columbia, Canada: ACM, 2011, pág. 87. ISBN: 978-1-4503-0971-4. DOI: 10.1145/2037715.2037813. URL: http://doi.acm.org/10.1145/2037715.2037813 (véase página 76).

[DV09] Julia Dmitrieva y Fons J. Verbeek. "Node-Link and Containment Methods in Ontology Visualization". En: *Proceedings of the 5th International Workshop on OWL: Experiences and Directions (OWLED 2009), Chantilly, VA, United States, October 23-24, 2009*. Ed. por Rinke Hoekstra y Peter F. Patel-Schneider. Vol. 529. CEUR Workshop Proceedings. CEUR-WS.org, 2009. DOI: http://ceur-ws.org/Vol-529/owled2009_submission_9.pdf (véase página 76).

[DW14] Tuan Nhon Dang y Leland Wilkinson. "ScagExplorer: Exploring Scatterplots by Their Scagnostics". En: *Pacific Visualization Symposium (PacificVis), 2014 IEEE*. IEEE, 2014, págs. 73-80 (véase página 59).

[Dal+95] Fergus Daly, David J. Hand, Chris Jones, Daniel Lunn y Kevin McConway. *Elements of Statistics*. Pearson Education.Limite, 1995 (véase página 18).

[DiB+92] David DiBiase, Alan M. MacEachren, John B. Krygier y Catherine Reeves. "Animation and the Role of Map Design in Scientific Visualization". En: *Cartography and Geographic Information Systems* 19.4 (1992), págs. 201-214, 265-266 (véase página 39).

[Don+88] A.W. Donoho, D.L. Donoho y M. Gasko. "MacSpin: Dynamic Graphics on a Desktop Computer". En: *Computer Graphics and Applications, IEEE* 8.4 (1988), págs. 51-58 (véase página 35).

[Du+99] Qiang Du, Vance Faber y Max Gunzburger. "Centroidal Voronoi Tessellations: Applications and Algorithms". En: *SIAM Review* 41.4 (1999), págs. 637-676 (véase página 116).

[EK00] Stephen G. Eick y Alan F. Karr. *Visual Scalability*. Inf. téc. National Institute of Statistical Sciencies, 2000 (véase páginas 1, 9, 17).

[Eic00] Stephen G. Eick. "Visual Discovery and Analysis". En: *IEEE Transactions on Visualization and Computer Graphics* 6.1 (2000), págs. 44-58. ISSN: 1077-2626. DOI: http://dx.doi.org/10.1109/2945.841120 (véase páginas 1, 18).

[Eic05] Stephen G. Eick. "Scalable Network Visualization". En: ed. por Charles D. Hansen y Christopher R. Johnson. Elsevier Academic Press, 2005. Cap. 42, págs. 819-829. ISBN: 0-12-387582-X (véase páginas 1, 10, 11).

[Eic96] Stephen G. Eick. "Aspects of Network Visualization". En: *IEEE Computer Graphics and Applications* 16.2 (1996), págs. 69-72. ISSN: 0272-1716. DOI: http://doi.ieeecomputersociety.org/10.1109/38.486685 (véase páginas 1, 14).

[Esc+11] Sebastian Escarza, Martin L. Larrea, Dana K. Urribarri, Silvia M. Castro y Sergio R. Martig. "Integrating Semantics into the Visualization Process". En: *Scientific Visualization: Interactions, Features, Metaphors*. Ed. por Hans Hagen. Vol. 2. Dagstuhl Follow-Ups. Dagstuhl, Germany: Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2011, págs. 92-102. ISBN: 978-3-939897-26-2. DOI: http://dx.doi.org/10.4230/DFU.Vol2.SciViz.2011.92. URL: http://drops.dagstuhl.de/opus/volltexte/2011/3304 (véase página 32).

[FD05] Michael Friendly y Daniel Denis. "The early origins and development of the scatterplot". En: *Journal of the History of the Behavioral Sciences* 41.2 (2005), págs. 103-130 (véase página 36).

[Fre+02] Carla M. D. S. Freitas, Paulo R. G. Luzzardi, Ricardo A. Cava, Marco Winckler, Marcelo S. Pimenta y Luciana P. Nedel. "On evaluating information visualization techniques". En: *Proceedings of the Working Conference on Advanced Visual Interfaces*. AVI '02. Trento, Italy: ACM, 2002, págs. 373-374. ISBN: 1-58113-537-8. DOI: 10.1145/1556262.1556326. URL: http://doi.acm.org/10.1145/1556262.1556326 (véase páginas 26, 60).

[Fri95] Michael Friendly. *Exploratory and Graphical Methods of Data Analysis*. SCS Short Course. 1995. URL: http://www.datavis.ca/courses/eda/ (véase páginas 18, 42).

[Fua+99] Ying-Huey Fua, Matthew O. Ward y Elke A. Rundensteiner. "Hierarchical parallel coordinates for exploration of large datasets". En: *VIS '99: Proceedings of the conference on Visualization '99*. Ed. por Bernd Hamann David S. Ebert Markus H. Gross. San Francisco, California, United States: IEEE Computer Society Press, 1999, págs. 43-50 (véase página 21).

[HC04] Jeffrey Heer y Stuart K. Card. "DOITrees Revisited: Scalable, Space-Constrained Visualization of Hierarchical Data". En: *Advanced Visual Interfaces (AVI)* (2004), págs. 421-424 (véase página 16).

[HJ05] Charles D. Hansen y Christopher R. Johnson, eds. *The Visualization Handbook*. Elsevier Academic Press, 2005. ISBN: 0-12-387582-X.

[Hao+10] Ming C. Hao, Umeshwar Dayal, Ratnesh K. Sharma, Daniel A. Keim y Halldór Janetzko. "Visual analytics of large multidimensional data using variable binned scatter plots". En: *Visualization and Data Analysis*. 2010, págs. 753006-753006-11. DOI: 10.1117/12.840142. URL: +http://dx.doi.org/10.1117/12.840142 (véase página 18).

[Hau+02] Helwig Hauser, Florian Ledermann y Helmut Doleisch. "Angular Brushing of Extended Parallel Coordinates". En: *INFOVIS '02: Proceedings of the IEEE Symposium on Information Visualization*. Washington, DC, USA: IEEE Computer Society, 2002, págs. 127-134. ISBN: 0-7695-1751-X (véase página 22).

[Her+98] Ivan Herman, Maylis Delest y Guy Melançon. "Tree visualisation and navigation clues for Information Visualisation". En: (1998) (véase página 83).

[Hes+01] David Hestenes, Hongbo Li y Alyn Rockwood. "A universal model for conformal geometries of Euclidean, spherical and double-hyperbolic spaces". En: London, UK: Springer-Verlag, 2001. Cap. 4, págs. 77-104. ISBN: 3-540-41198-4 (véase página 109).

[ID90] A. Inselberg y B. Dimsdale. "Parallel coordinates: A tool for visualizing multidimensional geometry". En: *IEEE Visualization* (1990), págs. 361-378 (véase páginas 14, 18).

[JH13] Mikkel R. Jakobsen y Kasper Hornbæk. "Interactive Visualizations on Large and Small Displays: The Interrelation of Display Size, Information Space, and Scale". En: *IEEE Transactions on Visualization and Computer Graphics* 19.12 (dic. de 2013), págs. 2336-2345. ISSN: 1077-2626. DOI: 10.1109/TVCG.2013.170. URL: http://dx.doi.org/10.1109/TVCG.2013.170 (véase página 1).

[JS03] J. A. Jacko y A. Sears, eds. *The Human–Computer Interaction Handbook: Fundamentals, Evolving Technologies and Emerging Applications*. Lawrence Erlbaum Associates, Inc., 2003.

[JS91] Brian Johnson y Ben Shneiderman. "Tree-Maps: A space-filling approach to the visualization of hierarchical information structures". En: *Visualization'91: Proceedings of the IEEE Conference on Visualization*. Ed. por Gregory M. Nielson y Larry Rosenblum. San Diego, CA, USA: IEEE Computer Society, 1991, págs. 284-291. ISBN: 0818622458. DOI: 10.1109/VISUAL.1991.175815 (véase páginas 73, 74).

[Jen08] Jeff S. Jenness. *Calculating areas and centroids on the sphere*. Poster presented at Arizona/New Mexico Wildlife Society Meeting. Albuquerque, New Mexico, USA (2008) and 28th Annual ESRI International User Conference. San Diego, California, USA. 2008 (véase página 116).

[Kei+10] Daniel A. Keim, Ming C. Hao, Umeshwar Dayal, Halldor Janetzko y Peter Bak. "Generalized Scatter Plots". En: *Information Visualization* 9.4 (2010), págs. 301-311 (véase páginas 37, 42).

[Kei00] Daniel A. Keim. "Designing Pixel-Oriented Visualization Techniques: Theory and Applications". En: *IEEE Transactions on Visualization and Computer Graphics* 6.1 (2000), págs. 59-78. ISSN: 1077-2626. DOI: http://dx.doi.org/10.1109/2945.841121 (véase página 1).

[Lam+95] John Lamping, Ramana Rao y Peter Pirolli. "A focus+context technique based on hyperbolic geometry for visualizing large hierarchies". En: *CHI '95: Proceedings of the SIGCHI conference on Human factors in computing systems*. Denver, Colorado, United States: ACM Press/Addison-Wesley Publishing Co., 1995, págs. 401-408 (véase páginas 76, 101).

[Lar+09] Martin Larrea, Dana Urribarri, Silvia Castro y Sergio Martig. "Spherical Layout implementation using Centroidal Voronoi Tessellations". En: *Journal of Computing* 1.1 (2009). `http://arxiv1.library.cornell.edu/pdf/0912.3974`, págs. 81-86. URL: `http://arxiv1.library.cornell.edu/pdf/0912.3974` (véase páginas 6, 77, 113).

[Lar06] Martín L. Larrea. "Diagramado Esférico". Tesis de lic. Universidad Nacional del Sur, 2006 (véase página 76).

[Lar10] Martín L. Larrea. "Visualización Basada en Semántica". Tesis doct. Universidad Nacional del Sur, 2010 (véase página 32).

[Lig+09] Peter Liggesmeyer, Jens Heidrich, Jürgen Münch, Robert Kalcklösch, Henning Barthel y Dirk Zeckzer. "Visualization of Software and Systems as Support Mechanism for Integrated Software Project Control". En: *Human-Computer Interaction. Novel Interaction Methods and Techniques: Proceedings of the HCI International*. Ed. por Julie A. Jacko. Lecture Notes in Computer Science. San Diego, CA, USA: Springer, 2009, págs. 846-855. ISBN: 9783642025761. DOI: `10.1007/978-3-642-02574-7_94` (véase página 75).

[MB05] Michael McGuffin y Ravin Balakrishnan. "Interactive visualization of genealogical graphs". En: *Proceedings of InfoVis 2005 - the IEEE Symposium on Information Visualization* (2005), págs. 17-24 (véase página 17).

[MD12] Abbe Mowshowitz y Matthias Dehmer. "Entropy and the Complexity of graphs revisited". En: *Entropy* 3.14 (2012), págs. 559-570 (véase página 101).

[MM02] W.L. Martínez y A.R. Martínez. *Computational statistics handbook with MATLAB*. Chapman and Hall/CRC Computer Science and Data Analysis Series. Chapman & Hall/CRC, 2002. ISBN: 9781584882299. URL: `http://books.google.com.ar/books?id=my-i9VNzCfAC` (véase página 18).

[MW95] Allen R. Martin y Matthew O. Ward. "High Dimensional Brushing for Interactive Exploration of Multivariate Data". En: *IEEE Conf. on Visualization '95*. 1995, págs. 271-278 (véase página 22).

[Mar+03] Sergio Martig, Silvia Castro, Pablo Fillottrani y Elsa Estevez. "Un Modelo Unificado de Visualización". En: *IX Congreso Argentino de Ciencias de la Computación*. La Plata, Argentina, 2003 (véase páginas 2, 29).

[Mil+97] Nancy Miller, Beth Hetzler, Grant Nakamura y Paul Whitney. "The need for metrics in visual information analysis". En: *Proceedings of the 1997 workshop on New paradigms in information visualization and manipulation*. NPIV '97. Las Vegas, Nevada, United States: ACM, 1997, págs. 24-28. ISBN: 1-58113-051-1. DOI: 10.1145/275519.275523. URL: http://doi.acm.org/10.1145/275519.275523 (véase página 26).

[Mun00] Tamara Munzner. "Interactive Visualization of Large Graphs and Networks". Tesis doct. Stanford University, 2000 (véase página 72).

[Mun97] Tamara Munzner. "H3: Laying Out Large Directed Graphs in 3D Hyperbolic Space". En: *IEEE Symposium on Information Visualization*. 1997, págs. 2-10 (véase páginas 72, 76, 77, 85).

[NL07] Roberto Navigli y Mirella Lapata. "Graph Connectivity Measures for Unsupervised Word Sense Disambiguation". En: Hyderabad, India, 2007, págs. 1683-1688 (véase página 101).

[PG88] R.M. Pickett y G.G. Grinstein. "Iconographic Displays For Visualizing Multidimensional Data". En: *Proceedings of the 1988 IEEE International Conference on Systems, Man, and Cybernetics, 1988.* Vol. 1. 1988, págs. 514-519 (véase páginas 36, 37, 39).

[PG92] Mark Phillips y Charlie Gunn. "Visualizing hyperbolic space: unusual uses of 4x4 matrices". En: *SI3D '92: Proceedings of the 1992 symposium on Interactive 3D graphics*. Cambridge, Massachusetts, United States: ACM, 1992, págs. 209-214. ISBN: 0-89791-467-8 (véase página 111).

[Pir+04] Harald Piringer, Robert Kosara y Helwig Hauser. "Interactive Focus+Context Visualization with Linked 2D/3D Scatterplots". En: *Proceedings of the Second International Conference on Coordinated & Multiple Views in Exploratory Visualization*. CMV '04. Washington, DC, USA: IEEE Computer Society, 2004, págs. 49-60. ISBN: 0-7695-2179-7. DOI: 10.1109/CMV.2004.11. URL: http://dx.doi.org/10.1109/CMV.2004.11 (véase páginas 37, 41).

[R] *The R Project for Statistical Computing*. URL: http://www.r-project.org/ (véase página 19).

[RS08] Adrian Rusu y Confesor Santiago. "Grid drawings of binary trees: An experimental study". En: *Journal of Graph Algorithms and Applications* 12.2 (2008), págs. 131-195 (véase página 100).

[RW] John Renze y Eric W. Weisstein. *"Scatter Diagram." From MathWorld–A Wolfram Web Resource.* URL: http://mathworld.wolfram.com/ScatterDiagram.html (véase página 35).

[Rat11] Bruce Ratner. *Statistical and Machine-Learning Data Mining: Techniques for Better Predictive Modeling and Analysis of Big Data*. 2.ª ed. CRC Press, 2011 (véase página 39).

[Rob+91] George G. Robertson, Jock D. Mackinlay y Stuart K. Card. "Cone Trees: animated 3D visualizations of hierarchical information". En: *CHI '91: Proceedings of the SIGCHI conference on Human factors in computing systems*. ACM, 1991, págs. 189-194. ISBN: 0-89791-383-3 (véase página 76).

[SG93] Ian Spence y Robert F. Garrison. "A Remarkable Scatterplot". En: *The American Statistician* 47.1 (1993), págs. 12-19 (véase página 40).

[Sch+11] Hans-Jorg Schulz, Steffen Hadlak y Heidrun Schumann. "The Design Space of Implicit Hierarchy Visualization: A Survey". En: *IEEE Transactions on Visualization and Computer Graphics* 17 (2011), págs. 393-411. ISSN: 1077-2626. DOI: http://doi.ieeecomputersociety.org/10.1109/TVCG.2010.79 (véase página 76).

[Sch11] Hans-Jörg Schulz. "Treevis.net: A Tree Visualization Reference". En: *Computer Graphics and Applications, IEEE* 31.6 (2011), págs. 11-15 (véase página 71).

[Sha48] Claude E. Shannon. "A Mathematical Theory of Communication". En: *The Bell System Technical Journal* 27.3 (1948), págs. 379-423 (véase página 101).

[Swa+98] Deborah F. Swayne, Dianne Cook y Andreas Buja. "XGobi: Interactive Dynamic Data Visualization in the X Window System". En: *Journal of Computational and Graphical Statistics* 7.1 (1998), págs. 113-130 (véase página 22).

[TC05] James J. Thomas y Kristin A. Cook, eds. *Illuminating the Path: The Research and Development Agenda for Visual Analytics*. National Visualization y Analytics Center, 2005.

[Tam13] Roberto Tamassia, ed. *Handbook of Graph Drawing and Visualization*. CRC Press, 2013 (véase páginas 99, 100).

[Tan+03] Yoichi Tanaka, Yoshihiro Okada y Koichi Niijima. "Treecube: Visualization Tool for Browsing 3D Multimedia Data". En: *IV'03: Proceedings of the International Conference on Information Visualisation*. London, UK: IEEE Computer Society, 2003, págs. 427-432. ISBN: 0769519881. DOI: 10.1109/IV.2003.1218020 (véase página 75).

[Tat+10] Andrada Tatu, Peter Bak, Enrico Bertini, Daniel Keim y Joern Schneidewind. "Visual quality metrics and human perception: an initial study on 2D projections of large multidimensional data". En: *Proceedings of the International Conference on Advanced Visual Interfaces*. AVI '10. Roma, Italy: ACM, 2010, págs. 49-56. ISBN: 978-1-4503-0076-6. DOI: 10.1145/1842993.1843002. URL: http://doi.acm.org/10.1145/1842993.1843002 (véase páginas 26-28, 60).

[Tou97] Thierry Toutin. "Qualitative Aspects of Chromo-Stereoscopy for Depth Perception". En: *Photogrammetric Engineering and Remote Sensing* 2.63 (1997), págs. 193-203 (véase página 41).

[Tuf01] Edward R. Tufte. *The Visual Display of Quantitative Information*. 2da edición (1era edición, 1983). Graphics Press, 2001 (véase página 25).

[Tuk77] John W. Tukey. *Exploratory Data Analysis*. Addison-Wesley, 1977 (véase página 39).

[Ung05] Abraham A. Ungar. *Analytic Hyperbolic Geometry. Mathematical Foundations and Applications*. World Scientific Publishing Co. Pte. Ltd, 2005 (véase página 109).

[Unw+06] Antony Unwin, Martin Theus y Heike Hofmann. *Graphics of Large Datasets: Visualizing a Million (Statistics and Computing)*. Secaucus, NJ, USA: Springer-Verlag New York, Inc., 2006. ISBN: 0387329064 (véase página 9).

[Urr+13] Dana K. Urribarri, Silvia M. Castro y Sergio R. Martig. "Gyrolayout: A Hyperbolic Level-of-Detail Tree Layout". En: *Journal of Universal Computer Science* 19.1 (2013), págs. 132-156. URL: `http://www.jucs.org/jucs_19_1/gyrolayout_a_hyperbolic_level` (véase páginas 3, 6, 72, 77, 113).

[Vli+06] Roel Vliegen, Jarke J. van Wijk y Erik-Jan van der Linden. "Visualizing Business Data with Generalized Treemaps". En: *IEEE Transactions on Visualization and Computer Graphics* 12.5 (2006), págs. 789-796. DOI: `10.1109/TVCG.2006.200` (véase páginas 74, 75).

[WF03] Martin Wattenberg y Danyel Fisher. "A model of multi-scale perceptual organization in information graphics". En: *Proceedings of the Ninth annual IEEE conference on Information visualization*. INFOVIS'03. Seattle, Washington: IEEE Computer Society, 2003, págs. 23-30. ISBN: 0-7803-8154-8. URL: `http://dl.acm.org/citation.cfm?id=1947368.1947377` (véase páginas 37, 40).

[WW99] Jarke J. van Wijk y Huub van de Wetering. "Cushion Treemaps: Visualization of Hierarchical Information". En: *InfoVis'99: Proceedings of the IEEE Symposium on Information Visualization*. Ed. por Daniel Keim y Graham Wills. San Francisco, CA, USA: IEEE Computer Society, 1999, págs. 73-78. ISBN: 0769504310. DOI: `10.1109/INFVIS.1999.801860` (véase página 74).

[Wat99] Martin Wattenberg. "Visualizing the stock market". En: *CHI'99: Extended abstracts of the SIGCHI conference on Human Factors in Computing Systems*. Pittsburgh, PA, USA: ACM Press, 1999, págs. 188-189. ISBN: 1581131585. DOI: `10.1145/632716.632834` (véase página 74).

[Weg95] E. Wegman. "Huge Data Sets and the Frontiers of Computational Feasibility". En: *Journal of Computational and Graphical Statistics* 4 (1995), págs. 281-295 (véase páginas 1, 9).

[Wet03] Kai Wetzel. *Pebbles – using Circular Treemaps to visualize disk usage*. `http://lip.sourceforge.net/ctreemap.html`. retrieved 26-APR-2010. 2003 (véase página 74).

[Wic09] Hadley Wickham. *ggplot2: elegant graphics for data analysis*. Springer New York, 2009. ISBN: 978-0-387-98140-6. URL: http://had.co.nz/ggplot2/book (véase páginas 37, 41).

[Wil+05] Leland Wilkinson, Anushka Anand y Robert Grossman. "Graph-Theoretic Scagnostics". En: *Proceedings of the Proceedings of the 2005 IEEE Symposium on Information Visualization*. INFOVIS '05. Washington, DC, USA: IEEE Computer Society, 2005, págs. 21-. ISBN: 0-7803-9464-x. DOI: 10.1109/INFOVIS.2005.14. URL: http://dx.doi.org/10.1109/INFOVIS.2005.14 (véase página 58).

[Wil05] Leland Wilkinson. *The Grammar of Graphics*. Statistics y Computing. Springer, 2005 (véase página 41).

[Wil97] G. J. Wills. "Nicheworks - interactive visualization of very large graphs". En: *Graph Drawing'97 Conference Proceedings*. Springer-Verlag, 1997 (véase página 19).

[YPF00] Julie Yang-Pelaez y Woodie C. Flowers. "Information Content Measures of Visual Displays". En: *Information Visualization, IEEE Symposium on* 0 (2000), pág. 99. ISSN: 1522-404X. DOI: http://doi.ieeecomputersociety.org/10.1109/INFVIS.2000.885096 (véase página 57).

[Yos07] Beth Ann Yost. "The Visual Scalability of Integrated and Multiple View Visualizations for Large, High Resolution Displays". Tesis doct. Virginia Polytechnic Institute y State University, 2007 (véase página 1).

[Zha+04] Kaidi Zhao, Bing Liu, Thomas M. Tirpak y Andreas Schaller. "V-Miner: using enhanced parallel coordinates to mine product design and test data". En: *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*. KDD '04. Seattle, WA, USA: ACM, 2004, págs. 494-502. ISBN: 1-58113-888-1. DOI: http://doi.acm.org/10.1145/1014052.1014108. URL: http://doi.acm.org/10.1145/1014052.1014108 (véase página 18).

[ADV10] ADVIZOR Solutions, Inc. *ADVIZOR 5.6*. 2010. URL: http://www.advizorsolutions.com/ (véase página 18).