Explorando Sistemas Dinámicos 3D no Lineales

Pablo Haramburu
Universidad de Buenos Aires
ph0u@dc.uba.ar

Claudio Delrieux*
Universidad Nacional del Sur y Universidad de Buenos Aires
claudio@acm.org

Palabras clave: visualización científica, visualización de sistemas dinámicos, ecuaciones diferenciales, exploración interactiva.

1. Objetivos de la linea de investigación

La finalidad de la Visualización Científica es ayudar en la comprensión y el análisis de determinados problemas científicos. Un modelo matemático utilizado muy frecuentemente para la representación de problemas reales en muchas disciplinas es el de los *sistemas dinámicos*. Es posible encontrarlos en ecología, electrónica, mecánica no lineal, dinámica de fluídos, matemática, economía, etc. En la mayoría de los casos, estos sistemas no son resolubles en forma analítica, por lo que su adecuada comprensión solo puede realizarse por medio de simulaciones computacionales, las cuales se representan de un modo natural, eficiente y más agradable por medios gráficos.

Los modelos matemáticos de sistemas dinámicos son variados: funciones iteradas, ecuaciones diferenciales ordinarias, autómatas, campos vectoriales, dinámica de fluídos, etc. Nosotros por el momento nos concentramos en sistemas dinámicos continuos representados mediante ecuaciones diferenciales ordinarias, autónomas en un primer paso. Campos vectoriales o flujo pueden ser considerados equivalentes, pero inducen puntos de vista levemente diferentes en la literatura de visualización científica.

Numerosas técnicas han sido propuestas para visualizar sistemas dinámicos [8]. Principalmente, basadas en trayectorias, en LIC, y en propiedades derivadas, como la topología o la vorticidad. Nuestra línea de investigación tiene como base un framework integrador (sección 2), y la búsqueda e implementación de técnicas para explorar sistemas dinámicos, especialmente los sistemas no lineales tridimensionales.

2. Framework

Son ya bien conocidas las múltiples técnicas de visualización de sistemas dinámicos (ver por ejemplo [1, 2, 3, 4, 5, 9, 10, 11] Frecuentemente los resultados de aplicar tales técnicas se publican aisladamente. Nuestra propuesta para integrar la presentación y manipulación de tales resultados parte de la analogía con el software de diseño asistido por computadora (CAD) y otros como el software de modelado y animación o de edición de imágenes (raster o vectoriales). Por ejemplo, un CAD permite manipular (ABM, reorganizar, ocultar, animar/parametrizar,

*Parcialmente financiado por la SECYT-UNS

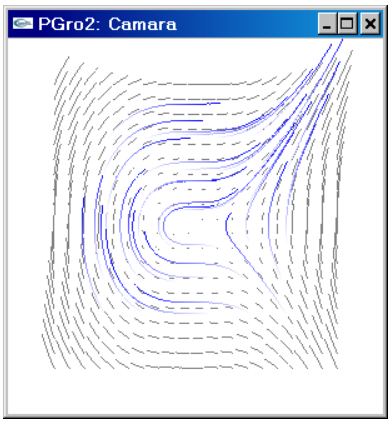

Figura 1: Una vista del sistema $(x', y') = (y, x^2)$.

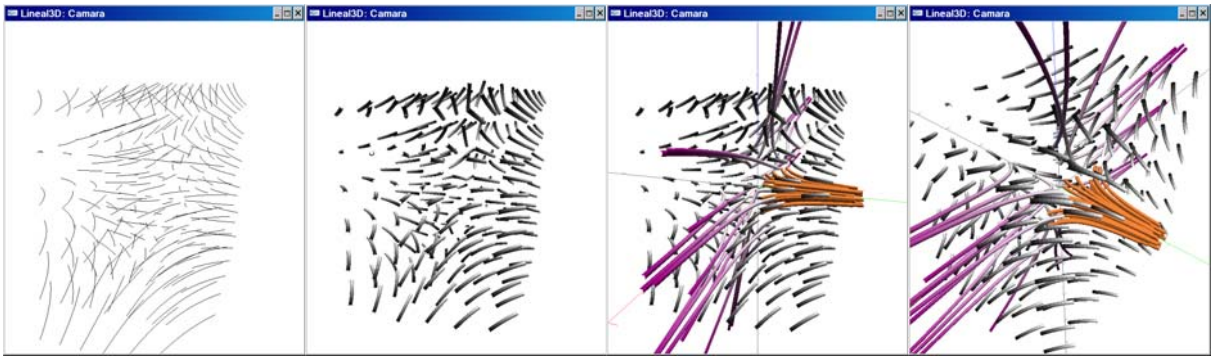

Figura 2: Ejemplo de una secuencia de vistas (*exploraciones*) del sistema lineal $(x', y', z') = (x, -y + 0{,}3z, 0{,}5z)$, con creciente calidad perceptual: trayectorias representadas por curvas; trayectorias representadas por tubos con iluminación y gradiente de color indicando el tiempo (blanco es el pasado); múltiples *sondas* y ejes de coordenadas, mejor marco de referencia; e interactividad, mejora la percepción tridimensional [12] (aquí otra vista).

etc.) diversas clases de objetos, en múltiples instancias. Sin embargo, no buscamos construir un "mundo" nuevo sino explorar un "mundo" preexistente, la ecuación diferencial o campo vectorial de interés.

Nuestro framework propone considerar a las diversas técnicas de visualización como *sondas* que se "lanzan" dentro de un mundo desconocido para explorarlo. Una sonda produce información sobre el objeto de estudio (la ecuación diferencial o campo vectorial), representada por una imagen, texto, animación, e incluso sonido. Por ejemplo, dibujar pequeñas flechas en una región del plano sería el resultado generado por una sonda de cierta clase. Otro ejemplo sería utilizar una textura y generar una visualización del espacio de fases mediante LIC [1].

En este proyecto buscamos dar soporte a exploración interactiva, al estilo de los sistemas de CAD, considerando mínimamente funciones como inserción y borrado de sondas y modificación de sus parámetros, incluyendo posición, orientación, tamaño y color cuando correspondan; organización y reorganización de sondas según criterios, por ejemplo en layers, en estructuras jerárquicas (sondas compuestas, relación padre-hijo; por ejemplo "a lo largo de una trayectoria"), por clase, etc.; mostrar y ocultar a voluntad; controlar, lo más interactivamente posible, los parámetros propios de cada sonda (los que dependen de la clase); múltiples vistas. Si bien no hay consenso en que las aplicaciones tipo CAD tengan actualmente la mejor solución como interfaz humano-computadora, también es cierto que, al estar tan difundidas, cualquier usuario de computadora está familiarizado con ellas, de modo que el ciclo ver-pensar-actuar resulta inmediato, reduciéndose el tiempo de aprendizaje.

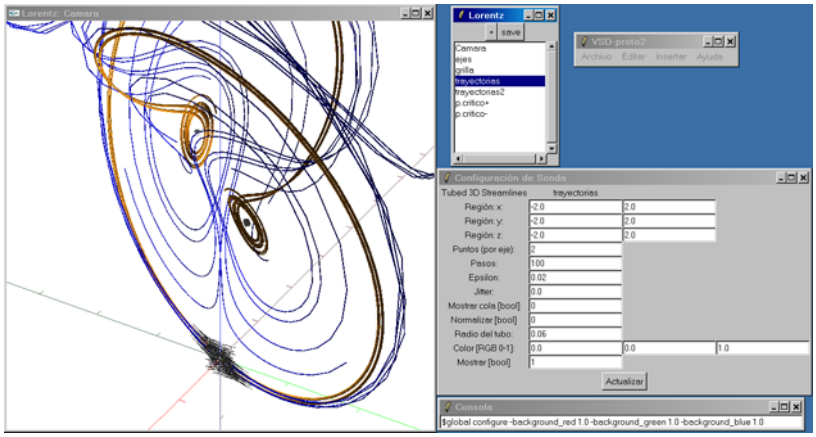

Figura 3: El prototipo 2 mostrando una vista de la ecuación de Lorentz, $(x', y', z') = (10(y - x), 28x - y - xz, xy - \frac{8}{3}z)$. Las trayectorias naranja, más gruesas, tienen más precisión que las azules, más delgadas (esto muestra la sensibilidad del sistema y el error de cálculo). Dos puntos (esferas blanco-grisáceas) sirven de mojón para ver la ubicación de dos puntos críticos del sistema.

Una sonda como "trayectorias" puede descomponerse en un generador de puntos o muestras (que se toman como condiciones iniciales, cada uno de una nueva trayectoria), un método de integración, y una técnica de representación. Esto también influencia la interfaz con el usuario (UI). El control de parámetros de una sonda puede presentarse en forma agrupada y descompuesta en los parámetros de los objetos que componen dicha sonda. También es posible recombinar (reutilizar).

Con este framework resulta razonable incluir otros objetos que no son sondas y darles similar tratamiento. Una clase amplia es la de los objetos geométricos: ejes de coordenadas (pares, ternas) proporcionan ubicación, orientación y tamaño (medida); puntos sirven de mojones, y se les puede adjuntar un cartel con las coordenadas u otras propiedades, como la magnitud escalar del campo, lo cual es un ejemplo de una sonda que produce un resultado de texto (que luego se grafica); textos o carteles para etiquetar otros objetos; objetos geométricos provenientes del problema de estudio (usual en problemas de fluidos, no tanto en matemática). Contar con esto permite al usuario descargar información de su memoria de corto plazo (o memoria de trabajo).

3. Trabajo realizado

Hemos implementado algunos de estos conceptos en pequeños prototipos. En los prototipos 0 y 1 probamos algunas ideas sobre la técnica de trayectorias (*streamlines*) en 3D: si las trayectorias son tubos y no líneas, los efectos de iluminación y oclusión mejoran la percepción tridimensional; y también un esquema de colores para ayudar a distinguir trayectorias (colorear variando a y b del espacio cromático Lab de manera suave). Estos prototipos permitían navegar la escena en forma precaria, pero carecían de UI interactiva, tomando los parámetros de la visualización desde archivo.

En el prototipo 2 incorporamos algunas mejoras: UI con interactividad rudimentaria (navegación, cuadros de diálogo); implementamos parcialmente el framework propuesto; apuntamos a plataformas Unix/Linux y Windows, para una potencial base de usuarios más amplia, de la cual obtener feedback. Utilizamos herramientas estándar: C++, OpenGL, Glut, Tcl/Tk, SWIG.

4. Conclusiones y trabajo futuro

Consideramos que el framework presentado tiene buenas chances de integrar e interrelacionar técnicas de visualización de sistemas dinámicos. Mencionamos seguidamente algunas cuestiones problemáticas que enfrentaremos a continuación.

4.1. Dimensiones

El prototipo 2 puede generar imágenes bi- y tridimensionales correspondientes a espacios de fase bi- y tridimensionales respectivamente. Es posible extender las técnicas de computación gráfica a cualquier cantidad de dimensiones, pero los resultados difícilmente sean comprensibles más allá de 4 o tal vez 5 dimensiones, incluso con sistemas estereoscópicos. El usuario se enfrentaría a la tarea de comprender la imagen mostrada al mismo tiempo que explorar un problema sobre el cual tiene interrogantes. Resulta entonces urgente encontrar e implementar técnicas para aliviar este problema, empezando por cortes o secciones, y siguiendo con técnicas como "worlds within worlds" de [6].

4.2. Derivados

Si el primer paso para visualizar un sistema dinámico es representar el campo vectorial asociado, y el segundo es obtener gráficos derivados, ya sea integrando (trayectorias, LIC) o buscando características (la topología: puntos críticos, ciclos límite, etc.), parece razonable que el siguiente paso sea obtener gráficos "aun más derivados". Por un lado, siguiendo esta línea, podemos considerar por ejemplo propiedades estadísticas, secciones de Poincaré y aplicación de Poincaré (Poincaré map) o aplicación de Lorentz (Lorentz map). Por otro, incorporar en consideración los *parámetros* del sistema, por ejemplo, mediante diagramas de bifurcaciones. El espacio a considerar en este caso es el producto cartesiano de los espacios de fase y de parámetros. Como este espacio usualmente tendrá más de tres dimensiones, es imperioso lo anterior (4.1). Queda por resolver cómo integrar "derivados" en el framework, es decir, si es posible evitar tener *explícitamente* (desde el punto de vista del usuario) una clases de gráfico por cada una de estas construcciones.

4.3. Problemas de escala

Muchos objetos geométricos involucrados en una exploración tienen uno o varios parámetros (tal vez implícitos) que dependen de la escala de dichos objetos. Por ejemplo, el diámetro de tubo del objeto que representa una streamline, la cantidad y distribución de dichos objetos, la cantidad marcas y separación entre ellas en ejes de coordenadas, etc. En algunos casos sería deseable contar con múltiples niveles de detalle (LOD), como la densidad de streamlines, de modo que aparezcan o desaparezcan según el nivel de zoom. Pareciera, en general, que cierta propiedad del objeto debiera ser aproximadamente constante en pantalla.

4.4. Texturas

El resultado de LIC es una textura, una imagen raster. Esto presenta el problema de integrarlo en una respresentación esencialmente vectorial (OpenGL), pues usarlo como textura sin más implica un filtrado que destruye el detalle y riqueza visual usado por la técnica de LIC para transmitir información.

4.5. Procesamiento en GPU

El hardware gráfico (GPU) experimentó una evolución formidable, pudiendo implementar una estructura de renderizado programable, con la capacidad de vincularse dinámicamente con la aplicación, y –además– la capacidad de ser programada interactivamente desde la aplicación [7]. Estas facilidades, más allá de la potencia inherente al hardware gráfico altamente paralelo, implican una forma completamente diferente y más efectiva de programar aplicaciones gráficas. Aprovechar estas ventajas implica abandonar las bibliotecas mayormente utilizadas para desarrollar aplicaciones gráficas y utilizarlas solo como mecanismo de comunicación entre la aplicación y la GPU. En aplicaciones tan exigentes tanto gráficamente como numéricamente como la que se describe aquí, las ventajas de procesar en la GPU son invalorables.

Referencias

[1] B. Cabral and L. Leedom. Imaging Vector Fields Using Line Integral Convolution. *ACM Computer Graphics (SIGGRAPH Proceedings)*, 25(3):263–270, 1993.

[2] Roger Crawfis, Nelson Max, and Barry Becker. Vector Field Visualization. *IEEE Computer Graphics and Applications*, 14(5):50–56, 1994.

[3] Thierry Delmarcelle and Lambertus Hesselink. Visualizing Second-Order Tensor Fields with Hypersrtreamlines. *IEEE Computer Graphics and Applications*, 13(4):25–33, 1993.

[4] Claudio Delrieux, Julián Dominguez, and Andrés Repetto. Towards a CLIC in Vector Field Visualization. In *Proceedings of the CISST 2001 Conference*, pages 695–702, CSREA Press, ISBN 1-892512-73-4, 2001.

[5] Claudio Delrieux, Julián Dominguez, and Andrés Repetto. Advanced Techniques for Real-time Flow Visualization. In *SPIE Proceedings Vol. 4716*, pages 375–385, The International Society for Optical Engineering Press, ISBN: 0-8194-4466-9, www.spie.org/web/abstracts/4700/4716.html, 2002.

[6] Steven K. Feiner and Clifford Beshers. Worlds within worlds: Metaphors for exploring n-dimensional virtual worlds. In Scott E. Hudson, editor, *User interface software and technology (UIST)*, pages 76–83. ACM Press, October 1990.

[7] Randima Fernando and Mark Kilgard. *The Cg Tutorial: The Definitive Guide to Programmable Real-Time Graphics*. Addison–Wesley, 2004.

[8] Helwig Hauser, Robert S. Laramee, and Helmut Doleisch. State-of-the-art report 2002 in flow visualization. Technical Report TR-VRVis-2002-003, VRVis Research Center, Vienna, Austria, Jan 2002.

[9] James Helman and Lambertus Hesselink. Visualizing Vector Field Topology in Fluid Flows. *IEEE Computer Graphics and Applications*, 11(3):36–46, 1991.

[10] J. J. van Wijk. Spot Noise: Texture Synthesis for Data Visualization. *ACM Computer Graphics*, 25(4):309–318, 1991.

[11] J. J. van Wijk. Flow Visualization with Surface Particles. *IEEE Computer Graphics & Applications*, 13(7):18–24, 1993.

[12] Colin Ware. *Information Visualization; Perception for design*. Morgan Kaufman, 2000.