# Métodos Interactivos de LIC para la Visualización de Diagramas de Fase

Claudio Delrieux, Julián Dominguez y Andrés Repetto
Departmento de Ingeniería Eléctrica - Universidad Nacional del Sur
Av. Alem 1253, (8000) Bahía Blanca, Argentina
e-mail:claudio@acm.org, jdoming@uns.edu.ar, repetto@uns.edu.ar

**Resumen**:

*Los métodos usuales para la visualización de diagramas de fase en sistemas dinámicos y flujos estacionarios parecen generar una disyuntiva de* calidad vs. interactividad. *En efecto, las técnicas que proveen posibilidades de interactividad (*streamlines*, por ejemplo) se caracterizan por producir resultados de calidad pobre, y requieren una excesiva supervisión para ajustarse a las propiedades específicas de un sistema en particular. Las técnicas de mayor calidad, como por ejemplo la LIC (*line integral convolution*) producen resultados aceptables sin intervención, pero con un costo computacional excesivo, aún en bajas resoluciones. Nuestro propósito en este trabajo es explorar la posibilidad de implementar métodos intermedios, en los cuales se combinen calidad, autonomía y tiempos interactivos, de manera de poder construir una herramienta de visualización de sistemas dinámicos implementada en la WEB, para usuarios en general, sin entrenamiento específico.*

Palabras Clave: Visualización Científica — Sistemas no Lineales y Caóticos — Streamlines — (LIC) Line Integral Convolution.

## 1 Introducción

La visualización de campos vectoriales, flujos, y diagramas de fase en sistemas dinámicos no lineales, constituye una de las áreas de mayor actividad en la visualización científica. Las técnicas desarrolladas para estos objetivos son de gran importancia teórica y experimental en diversas disciplinas, como por ejemplo dinámica de los fluídos, ecología, electrónica, mecánica no lineal, etc. Estas aplicaciones han ido cobrando importancia en la misma medida en que los sistemas dinámicos han sido incorporados y utilizados como modelo matemático. En la mayoría de los casos, estos sistemas no son integrables 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 y eficiente por medios gráficos.

En la literatura se han presentado varios métodos para la visualización de campos vectoriales (glifos, trayectorias de partículas, íconos, Streamlines, Hyperstreamlines, Spot Noise, LIC, etc.) [1, 2, 3, 15, 19, 20], de las cuales Streamlines y LIC son las más importantes y utilizadas. En flujos estacionarios, el método de las Streamlines o líneas de flujo coincide con la integración de trayectorias en sistemas dinámicos [12, 18]: se posiciona

una condición inicial (o *semilla*) en el espacio de fases, y se integra la ecuación diferencial por medio de diferenciales finitos. Los sucesivos puntos que se encuentran constituyen una *trayectoria* del diagrama de fases del sistema. Un problema importante es encontrar un conjunto representativo de trayectorias, las cuales reflejen todas las características relevantes del diagrama de fases del sistema (para obtener cubrimiento uniforme o para ubicar todos los puntos críticos del sistema). Este problema implica encontrar un conjunto adecuado de semillas, una longitud correcta para las trayectorias, y también un valor computacionalmente eficaz para el diferencial finito. Todos estos requisitos son directamente dependientes del sistema particular, por lo que la visualización adecuada de un diagrama de fases es necesariamente un proceso asistido por el usuario.

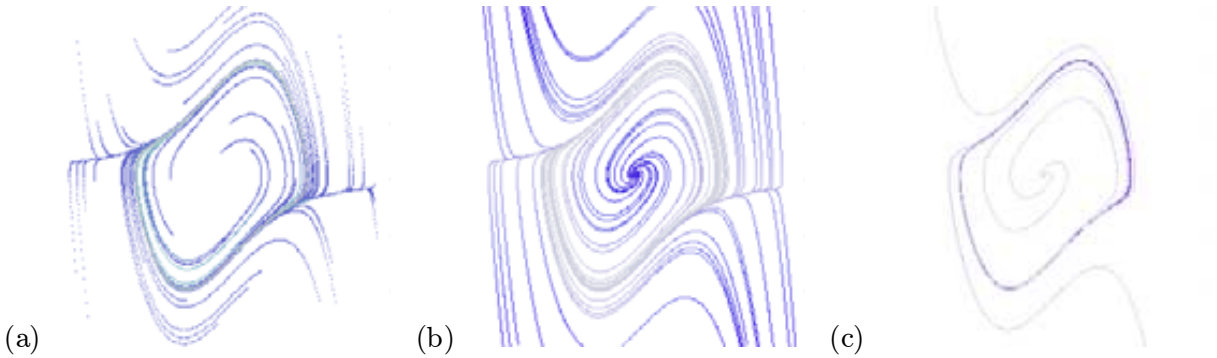
Por su parte, la LIC (Line Integral Convolution) [1] es un método basado en texturas, en el cual cada punto $p$ del espacio de fases se pinta con un color obtenido al efectuar una convolución con la textura de entrada a lo largo de una trayectoria que pasa por $p$. Como es posible ver, por su misma naturaleza, la LIC produce un cubrimiento uniforme del diagrama de fases, y todos los puntos críticos del sistema son adecuadamente representados. Por dicha razón, la LIC ha sido intensivamente utilizada en la visualización de flujos y campos vectoriales, y la mejora de sus posibilidades constituye un activo campo de investigación (por ejemplo, visualizar magnitud y dirección del flujo por medio de animación [13, 21], visualización de campos vectoriales 3D y flujos no estacionarios [13], optimización y reducción del costo computacional [11, 14, 17]).

Este último punto es sin duda el más importante, dado que la evaluación de una LIC tiene un costo computacional varios órdenes de magnitud mayor que una adecuada representación con streamlines. Por dicha razón es que en este trabajo investigamos la posibilidad de elaborar métodos para visualizar campos vectoriales que sean robustos y se acerquen a la calidad de la LIC, pero con tiempos de cómputo cercanos a los streamlines. El objetivo final es poder proveer un "servicio de visualización" basado en la WEB, donde la aplicación cliente le permita a un usuario genérico visualizar los diagramas de fase en sistemas dinámicos arbitrariamente elegidos.

# 2 Trabajo previo y características de los métodos

El método de las streamlines, como vimos, se basa en la representación de trayectorias computadas a partir de una condición inicial o "semilla", por medio de una integración numérica. Esta forma de proceder es muy eficaz, produciendo una visualización adecuada en fracciones de segundo. Para obtener un cubrimiento uniforme del diagrama de fases o para ubicar todos los puntos críticos del sistema, se requiere encontrar un conjunto representativo de trayectorias, lo cual implica encontrar las respectivas semillas, y el paso y la longitud de integración. Una manera automática de generar las semillas es posicionarlas uniformemente en el espacio de fases. De esa forma es posible ubicarlas con una posición y densidad arbitrarias. Usualmente esta técnica produce *aliasing* cuando la frecuencia espacial de posicionamiento interactúa con las características del diagrama de fases, por lo que es aconsejable utilizar una perturbación de Poisson [4].

La elección de un valor para el diferencial finito o factor de integración es también un problema, dado que un paso demasiado conservador puede producir *oversampling*, es decir, la generación de una excesiva cantidad de muestras por trayectoria, mayor que



(a) (b) (c)

Figura 1: Oscilador de van Der Pol. (a) Densidad de semillas 3%, (b) *idem* con *Path Search*, y (c) iteración inversa para visualizar el ciclo límite atractivo.

la cantidad de pixels en la misma. Por otro lado, un paso de integración muy grande producirá *undersampling*, es decir, una cantidad inadecuadamente baja de muestras por trayectoria, lo cual en streamlines se aprecia como una "desintegración" exagerada en la representación de la trayectoria (ver Fig. 1(a)). El valor adecuado es imposible de determinar estáticamente, aún en un mismo sistema dinámico, porque obviamente depende de la velocidad local del flujo. La única manera de asegurarse un paso óptimo en toda circunstancia consiste en utilizar un paso adaptativo, como se sugiere en [5].

La cantidad de iteraciones, junto con el paso de integración y la velocidad media en la trayectoria, son todos factores que inciden en la longitud final que tendrá la misma en el diagrama de fases. En muchas circunstancias la correcta apreciación de un campo de velocidades se logra visualizando un gran conjunto de trayectorias relativamente cortas ("colas de cometa" [8, 10]), y para brindar una buena idea de la dirección del flujo se pueden representar las trayectorias modificando la saturación (ver Fig. 1) [11]. Estas técnicas son directamente dependientes del sistema en particular, por lo que una visualización adecuada requiere de la asistencia del usuario.

El posicionamiento uniforme de las semillas no garantiza un cubrimiento satisfactorio. Ni siquiera garantiza que todos los puntos críticos del sistema dinámico se representen adecuadamente. La técnica de la saturación permite distinguir atractores de repeledores, así como ubicar ciclos límite y su estabilidad. Sin embargo el diagrama de fases quedará más cubierto en cercanías de atractores o ciclos límite atractivos, mientras que en cercanías de los repeledores el diagrama es más disperso. Una solución para estos problemas consiste en realizar la dinámica inversa del sistema a partir de la semilla, una cierta cantidad de iteraciones, para determinar si el punto está cerca de un repeledor. Esta técnica de "path-search" [6] permite lograr un cubrimiento más parejo del diagrama de fases, y además acelera la generación del diagrama de fases porque requiere integrar una menor cantidad de trayectorias (ver Fig. 1(b)).

Por su parte, los métodos basados en texturas realizan una integración más corta (típicamente se realizan menos de 100 iteraciones). La trayectoria generada no se grafica, sino que se utiliza para recorrer los texels correspondientes, integrando el color de los mismos para encontrar el color final del pixel [1]. Una mejora inmediata consiste en multiplicar, para cada pixel, el aporte de cada texel por medio de una misma funcion ponderadora o *kernel* de convolución de longitud L igual a la cantidad de pasos de integración. De esa manera se computa cada pixel de la imagen final. Por dicha razón, la LIC produce un cubrimiento uniforme del diagrama de fases, por lo que todos los pun-

tos críticos del sistema son satisfactoriamente representados. Sin embargo muchas veces resulta difícil identificarlos adecuadamente. No siempre se pueden distinguir atractores de repeledores, y resulta casi imposible determinar la existencia de trayectorias cerradas (focos, ciclos límite, trayectorias homoclínicas, ver [9]). La mejora de las posibilidades de la LIC constituye un activo campo de investigación [11, 13, 14, 17, 21], especialmente en lo que concierne a mejorar su costo computacional y la representación del sentido del flujo y las trayectorias cerradas.

Muchas de las mejoras mencionadas más arriba para las streamlines son directamente aplicables en la LIC, aunque, como veremos en la siguiente Sección, deben combinarse de una manera especial. En particular podemos mencionar que la LIC adaptativa es aproximadamente un orden de magnitud más veloz [5]. En la siguiente Sección presentaremos un método acumulativo para computar la LIC el cual, con una combinación de sembrado no uniforme, desaturación y *path search* produce una aceleración ulterior de otro orden de magnitud más. Por lo tanto, la velocidad de cómputo obtenida con estos métodos derivados de la LIC se acerca bastante bien a las posibilidades interactivas buscadas.

Por último, podemos mencionar un aspecto de importancia: la textura utilizada para la generación de la LIC. En efecto, la textura constituye un medio para introducir determinadas características geométricas y cromáticas en la imagen final, las cuales interactúan con la distribución geométrica de las trayectorias en el diagrama de fases. Por lo tanto, una elección ingeniosa de texturas es un medio adecuado para enfatizar o resaltar determinadas características que se busca visualizar en un campo vectorial arbitrario. Este aspecto ha sido poco estudiado hasta ahora. Por ejemplo en [16] se considera solamente la distribución espectral de diversos ruidos utilizados como textura. Un ruido blanco de alta frecuencia como textura de entrada permite resaltar los detalles finos en el campo vectorial, mientras que el mismo ruido filtrado con un pasabajos tiende a destacar la forma global del diagrama de fases. En [6, 8] se sugiere un conjunto de técnicas para generar texturas más adecuadas, las cuales luego pueden ser utilizadas para resaltar diversos aspectos de un determinado campo vectorial.

# 3 LIC acumulativo

El LIC acumulativo (*Cumulative LIC o CLIC*) [6, 8] surge de combinar las técnicas de desaturación de las trayectorias, y de posicionamiento no uniforme de las semillas en las streamlines. La idea consiste en acumular el color de los texels visitados en una LIC, en vez de computar una convolución (ver Fig. 2), e ir graficando el color acumulado en la imagen final. De esa forma, el cómputo del color final de un pixel es mucho más sencillo que con la LIC, pueden reutilizarse los cómputos parciales para encontrar los colores de los pixels vecinos, y cada trayectoria computada genera varios pixels en la imagen final. En general esta técnica permite bajar en promedio un orden de magnitud el tiempo de cómputo con calidades similares a la LIC original. Además el CLIC puede utilizarse con todas las demás técnicas mencionadas en la sección anterior (paso adaptativo, *path search*), con lo cual se obtiene un método cuyo tiempo de cómputo es muy cercano a los streamlines, pero con la calidad de la LIC. La base del algoritmo CLIC se muestra en la Tabla 1.

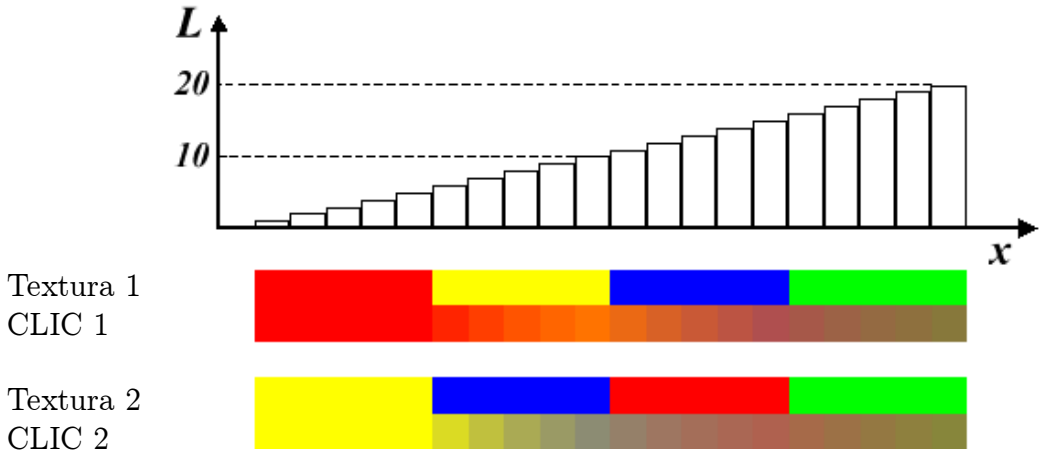



Figura 2: Representación gráfica (1D) del comportamiento del CLIC, aplicado a dos texturas. El sentido del flujo es de izquierda a derecha.

```
i := 1
ColorClic(x0, y0):= ColorTextura(x0, y0)
(x, y) := (x0, y0)
Mientras (i <= L) y (ε< e^Exp) y ((x, y) dentro del diagrama) hacer
  encontrar próximo pixel (x, y, x1, y1)
  si (ε< e^Exp) y ((x1, y1) dentro del diagrama) entonces
    ColorClic(x1, y1):= (i ColorTextura(x, y) + ColorTextura(x1, y1)) / (i + 1)
  i := i + 1
End.
```

Tabla 1: LIC acumulativo (CLIC).

Un detalle importante del CLIC es que –manejado cuidadosamente– permite utilizar kernels mucho mayores que la LIC convencional ($> 500$) sin un incremento dramático en el tiempo de cómputo. Aplicado en forma directa, sobre cada pixel de la figura final, produce resultados similares a una LIC de mismo kernel L, con tiempos sensiblemente menores (ver Fig. 3). El mejor aprovechamiento de la técnica consiste en efectuar un "sembrado" no uniforme en un determinado porcentaje de los pixels de la imagen de salida. Experimentalmente podemos mencionar que generando una trayectoria CLIC cada 5 o 10 pixels, se produce un cubrimiento casi total de la imagen de salida (dependiendo de cómo sea el diagrama de fases y del valor de L elegido). El cubrimiento de la imagen final puede mejorarse sensiblemente utilizando *path search*, y eventualmente se puede computar un segundo paso de CLIC para generar una trayectoria en cada pixel no visitado, obteniéndose de esa manera un cubrimiento total.

Es fácil ver, entonces, que aumentando la longitud L del kernel no aumenta necesariamente el tiempo de cómputo del CLIC. Un L mayor genera trayectorias más largas, con más pixels visitados, lo cual permite disminuir tanto la cantidad de trayectorias a recorrer como la cantidad de cómputo en el segundo paso. En la Fig. 4 comparamos todas estas mejoras graduales utilizando un kernel L=300. El algoritmo LIC original requiere 212 seg. para generar la imagen. El CLIC adaptativo sembrado en cada pixel requiere 2.5 seg. y el

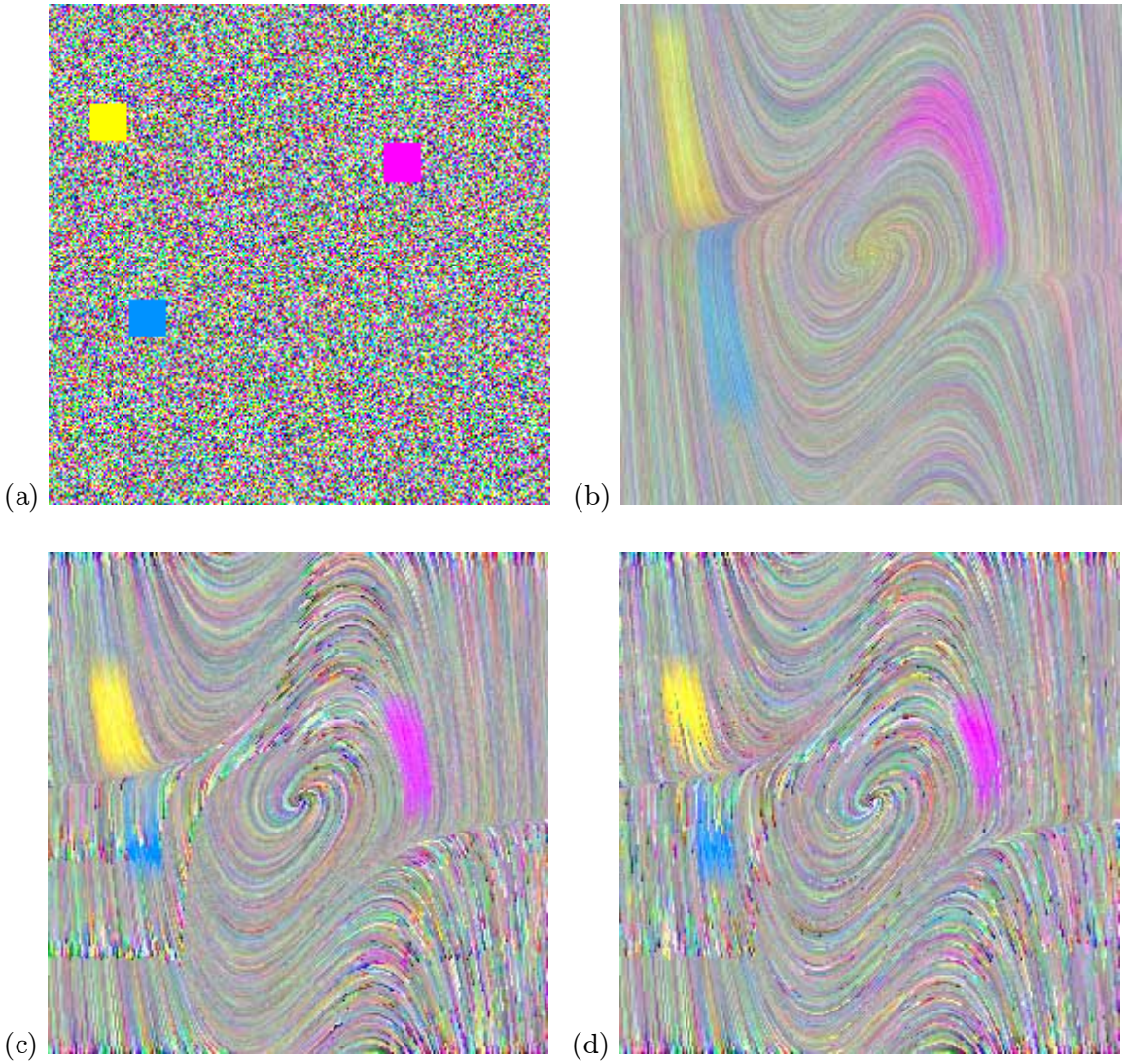


Figura 3: (a) Una textura de entrada, y (b) la LIC utilizando como sistema dinámico el mismo que en la Fig. 1. Con un kernel L=50 se requirieron t=20 seg. para obtener una imagen de 256×256, (c) utilizando CLIC en cada pixel (t=2.8 seg.), y (d) utilizando CLIC en uno cada 5 pixels (t=0.87 seg.).

CLIC adaptativo con *path search*, sembrado en el 10% de los pixels y con segundo paso, requiere t=0.65 seg.

Una observación detallada realizada a las imágenes resultantes puede mostrar que las trayectorias con L grandes van perdiendo saturación muy rápidamente, lo cual dificulta la visualización de las características geométricas del diagrama de fases. Una forma de solucionar este problema consiste en utilizar una función ponderadora lineal (de muy bajo costo computacional) que permita acumular el color a lo largo de la trayectoria con menor intensidad. De esa forma el factor $k$ o *carry* permite controlar la desaturación de las trayectorias (ver la línea correspondiente en la Tabla 1):

`ColorClic`$(x_1, y_1)$`:= (`$k$`*`$i$ `ColorTextura`$(x, y)$ + `ColorTextura`$(x_1, y_1)$`) / (`$k$`*`$i$+1).

En la Fig. 5 podemos ver el efecto del factor $k$ sobre el CLIC. Utilizamos el mismo sistema dinámico, con dos texturas diferentes. En la fila superior utilizamos una textura compuesta por líneas rectas de pendiente aleatoria, cuyo color depende de la pendien-

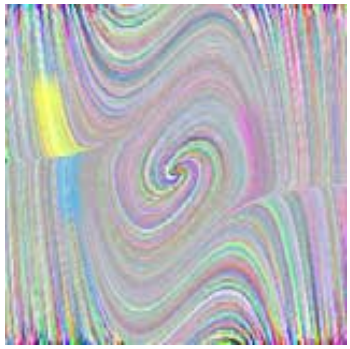
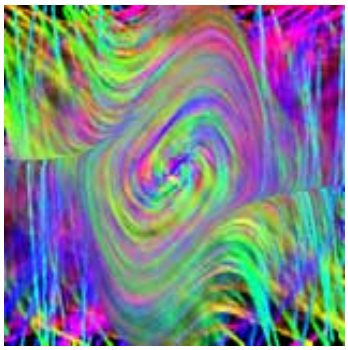
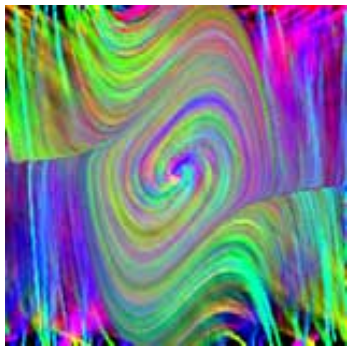
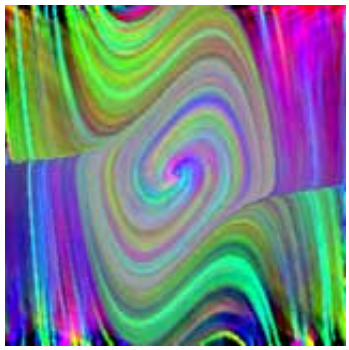
(a)  (b)  (c) 

Figura 4: (a) LIC original, L=300, t=212 seg. (b) CLIC adaptativo sembrado en cada pixel, L=300, t=2.5 seg. (c) CLIC adaptativo con *path search*, sembrado al 10% y con segundo paso, L=300, t=0.65 seg.

(a)  (b)  (c) 

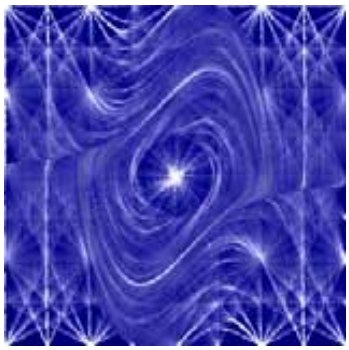
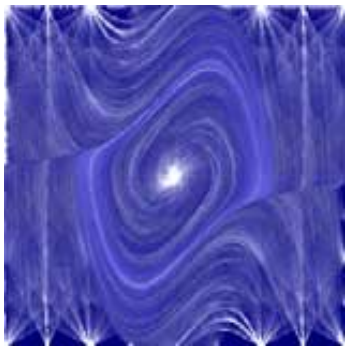
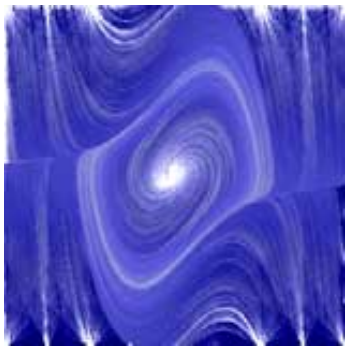
(d)  (e)  (f) 

Figura 5: Visualizaciones del diagrama de fases del oscilador de Van der Pol. La fila superior utiliza una textura diferente a la fila inferior. La columna izquierda se computó con $k = 0.2$, la central con $k = 0.5$ y la derecha con $k = 1$.

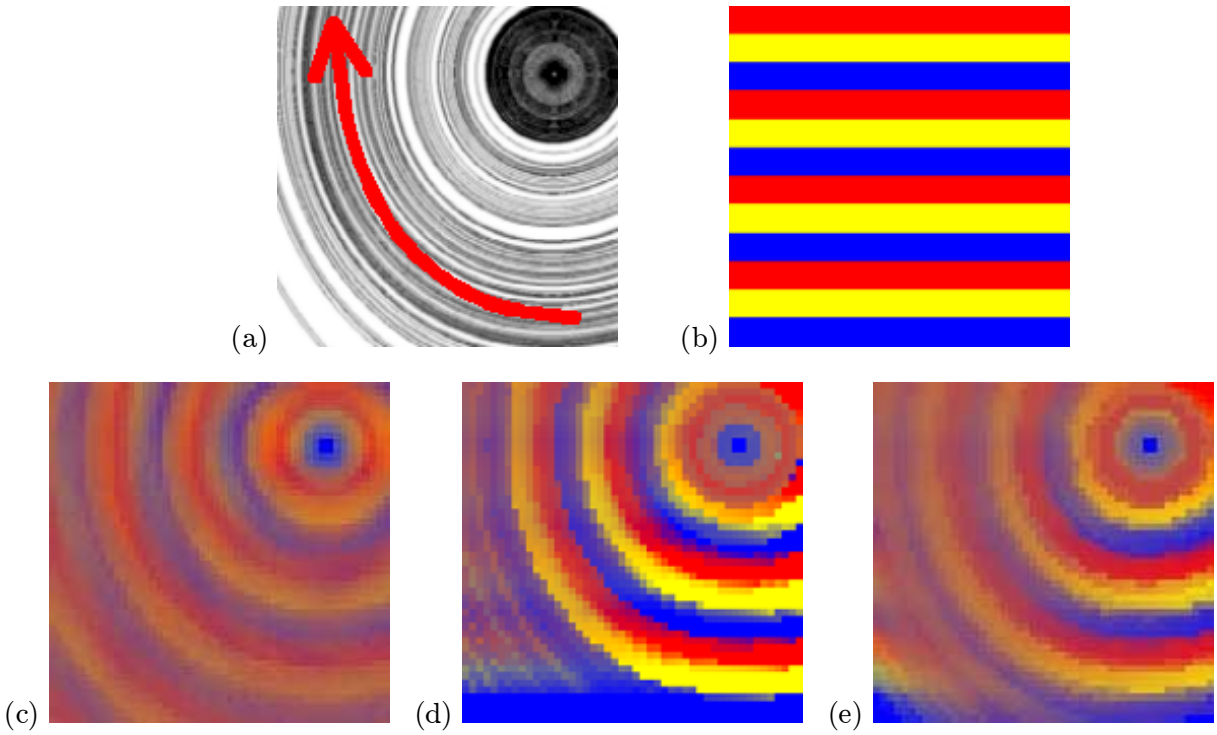



Figura 6: (a) Un campo vectorial, y (b) la textura utilizada. En (c) vemos la LIC respectiva, en (d) el CLIC y (e) el doble CLIC.

te. De esa forma la textura "sincroniza" con las líneas de flujo en cualquier dirección, haciendo que las mismas cambien de color cuando cambia la pendiente. Se observa que con menor $k$ la influencia de la textura original es mayor. En la fila inferior utilizamos una textura con una geometría más ordenada (líneas de un mismo color, con pendientes y ubicaciones uniformemente distribuidas). Esta textura enfatiza la ocurrencia de líneas de flujo con direcciones específicas. Se puede observar, sin embargo, que su coherencia geométrica interfiere un poco con el diagrama de fases, aún con valores altos de $k$. Estas visualizaciones sugieren que un método adecuado para visualizar las direcciones del flujo consiste en animar *la textura*. Por ejemplo, utilizar líneas paralelas en rotación permite mostrar en forma sucesiva las diferentes orientaciones de las líneas de flujo. Utilizar líneas de pendiente aleatoria con grosor variable permite ir mostrando de la baja frecuencia a la alta (o viceversa), etc. (ver [7]).

# 4 Doble CLIC

Como pudimos ver en la Sección anterior, el CLIC y sus mejoras asociadas nos permiten visualizar diagramas de fase en tiempos semejantes a las streamlines, con calidades semejantes a la LIC. Sin embargo, el método es nuevamente dependiente del sistema dinámico en particular y requiere la elección cuidadosa de los parámetros por parte del usuario para ser correctamente utilizado. Por dicha razón en esta Sección veremos de qué modo es posible dotar al CLIC de cierta autonomía para que produzca resultados adecuados en tiempos interactivos y en casi cualquier condición. Esto contribuye a nuestro propósito

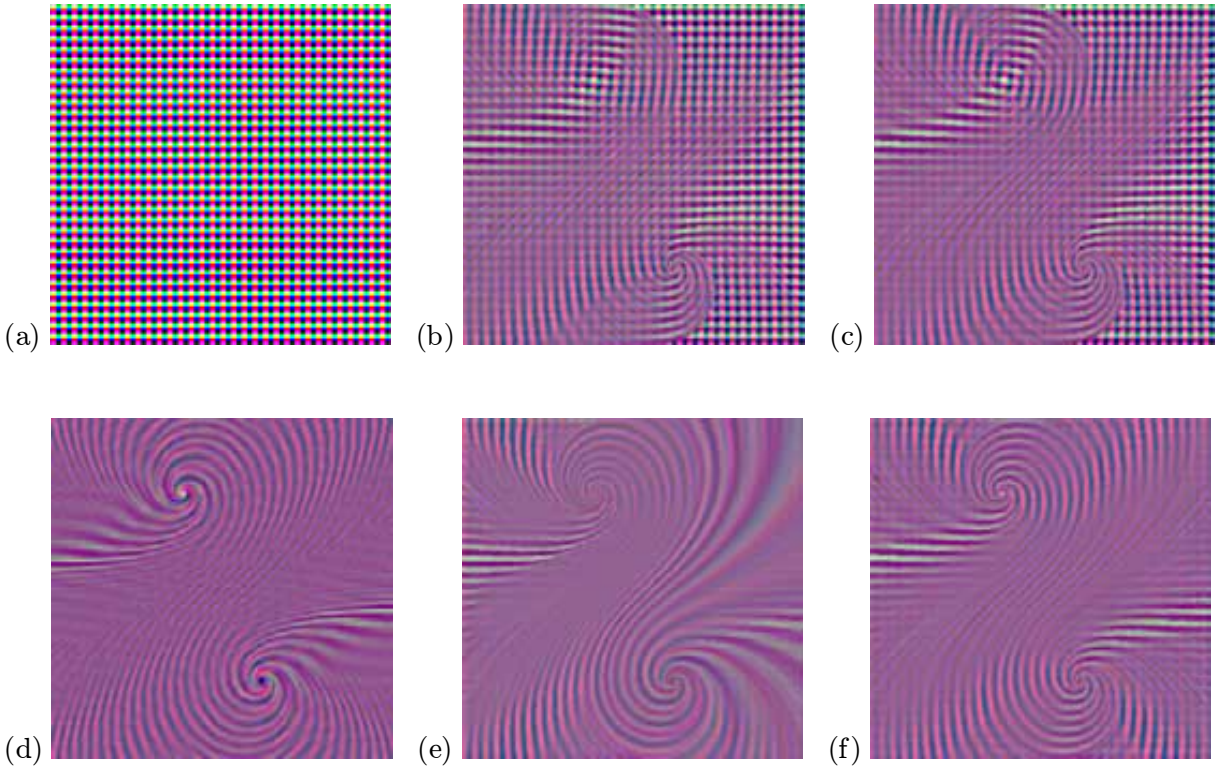



Figura 7: Tres evaluaciones del diagrama de fases. En (a) la textura, y en (d) la LIC original con L=50 y un tiempo de cómputo de 20 seg. En (b) el CLIC calculado "hacia atrás" (recorrido en sentido inverso) con k=0,05 y luego, en (e) computado hacia adelante con la imagen (b) tomada como textura, y un k=0,5. En ambos casos L=500. De la misma forma se procedió con (c) y (f), pero en ambos casos con k=0.1. Cada CLIC por separado insumió 0.35s., por lo que el tiempo total para el doble CLIC es de 0,7 seg.

final de lograr un sistema interactivo de propósito general para visualizar campos vectoriales arbitrarios.

Según se mencionara más arriba, la textura de entrada juega un rol preponderante en la calidad del resultado, y dado que el CLIC es notablemente económico para computar, la idea para lograr un método autónomo consiste en utilizar un CLIC (probablemente de baja calidad pero poco costo computacional) *como textura de entrada* para el cómputo del CLIC. Es decir, aplicamos dos veces el método. La primera transforma la textura original en una textura intermedia de mejor calidad, y la segunda produce la visualización final. De esa forma, el "doble CLIC" logra combinar la rapidez y autonomía de los streamlines con la calidad de la LIC. En la Fig. 6 podemos comparar, para un campo vectorial y una textura especialmente difícil, el efecto de aplicar dos veces el CLIC. De esa forma se logra un efecto visual de mayor semejanza al LIC original.

Esta forma de trabajar permite elegir diferentes valores de $k$ para realizar un CLIC inverso y luego hacia adelante (utilizando el primero como textura de entrada para el segundo). En la Fig. 7 se comparan tres visualizaciones del mismo diagrama de fases. En la Fig. 7(a) vemos la textura de entrada, y en la Fig. 7(d) la LIC original con L=50 y un tiempo de cómputo de 20 seg. En la Fig. 7(b) se muestra el CLIC recorrido en sentido inverso con k=0,05 y luego, en la Fig. 7(e) computado hacia adelante con la Fig. 7(b) tomada como textura, y un k=0,5. De igual modo se procedió en las Figs. 7(c) y (f), pero

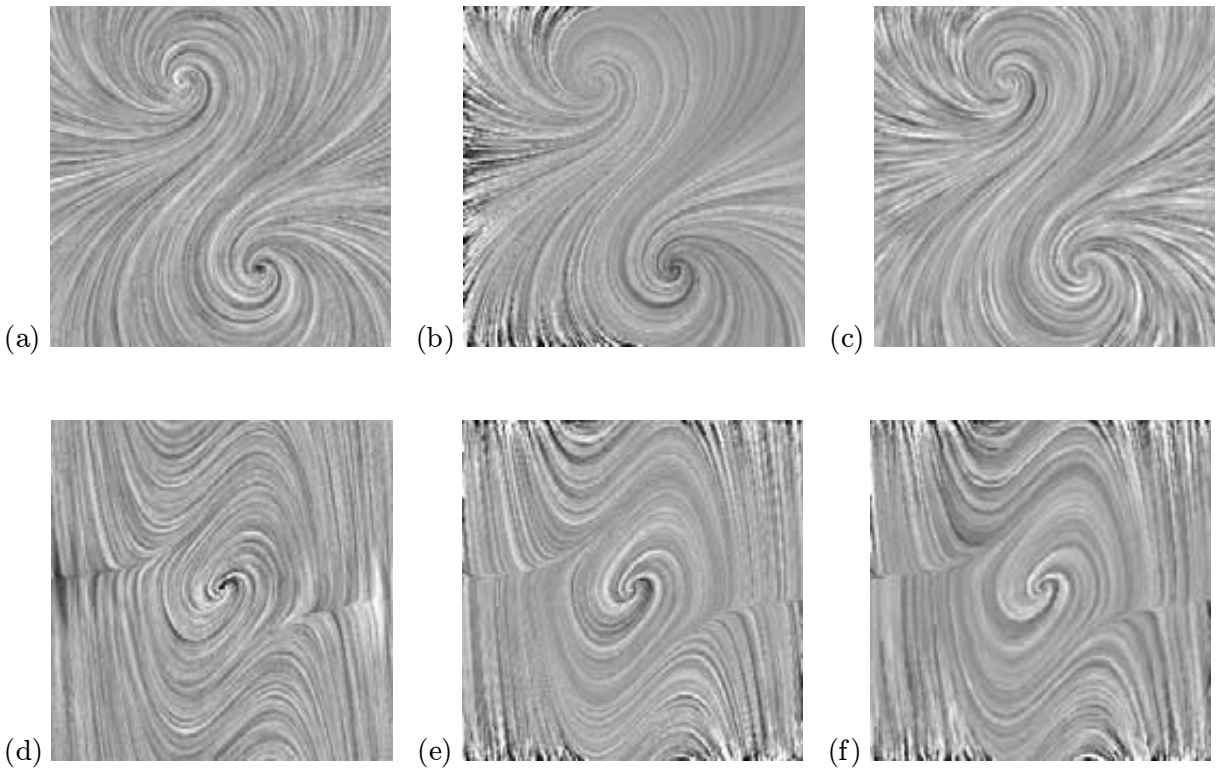



Figura 8: Vemos dos sistemas graficados a partir de la tradicional textura de ruido blanco y negro, con el fin de comparar los resultados. La primera fila fue hecha con el sistema de Julia y la segunda con el oscilador de van Der Pol. De izquierda a derecha vemos respectivamente la LIC original, el CLIC y el doble CLIC.

con k=0.1. tanto en sentido directo como inverso. En estas dos últimas evaluaciones se utilizó un kernel L=500, requiriendo cada CLIC un tiempo de 0.35s., por lo que el tiempo total para el doble CLIC es de 0,7 seg.

En la Fig. 8 utilizamos como textura de entrada el clásico ruido blanco (monocromático) en dos sistemas dinámicos diferentes, con la LIC original, el CLIC y el doble CLIC, a efectos de comparar los resultados obtenidos. Se observa que la visualización producida por los nuevos métodos (CLIC y doble CLIC) son de calidad semejante a la LIC, con una eficiencia computacional mucho mayor.

En la Fig. 9 vemos la interfase del CLIC Workbench, el sistema implementado para testear todos los métodos sugeridos en este trabajo. Podemos ver la ventana de edición de texturas de entrada, donde es posible seleccionar un tipo de textura y los parámetros involucrados para generarla. Algunas texturas son geométricamente ordenadas (rectas, círculos, discos, etc., otras generan elementos geométricos aleatoriamente posicionados (rectas, cuadrados, discos) y otras utilizan ruido. Los parámetros de estas texturas en general permiten ajustar la cantidad, tamaño y variación de los elementos geométricos, lo cual, como vimos, incide en el realce a características específicas del diagrama de fases. En la ventana de graficación se muestra la evaluación del sistema dinámico elegido, con el método de cómputo y los parámetros seleccionados (tamaño del kernel, paso de integración, paso fijo o adaptativo, muestreo uniforme o Poisson, etc.). El sistema dinámico puede elegirse de un conjunto predefinido de casos de estudio -con sus parámetros adecuados- o puede ingresarse por fórmula simbólica.

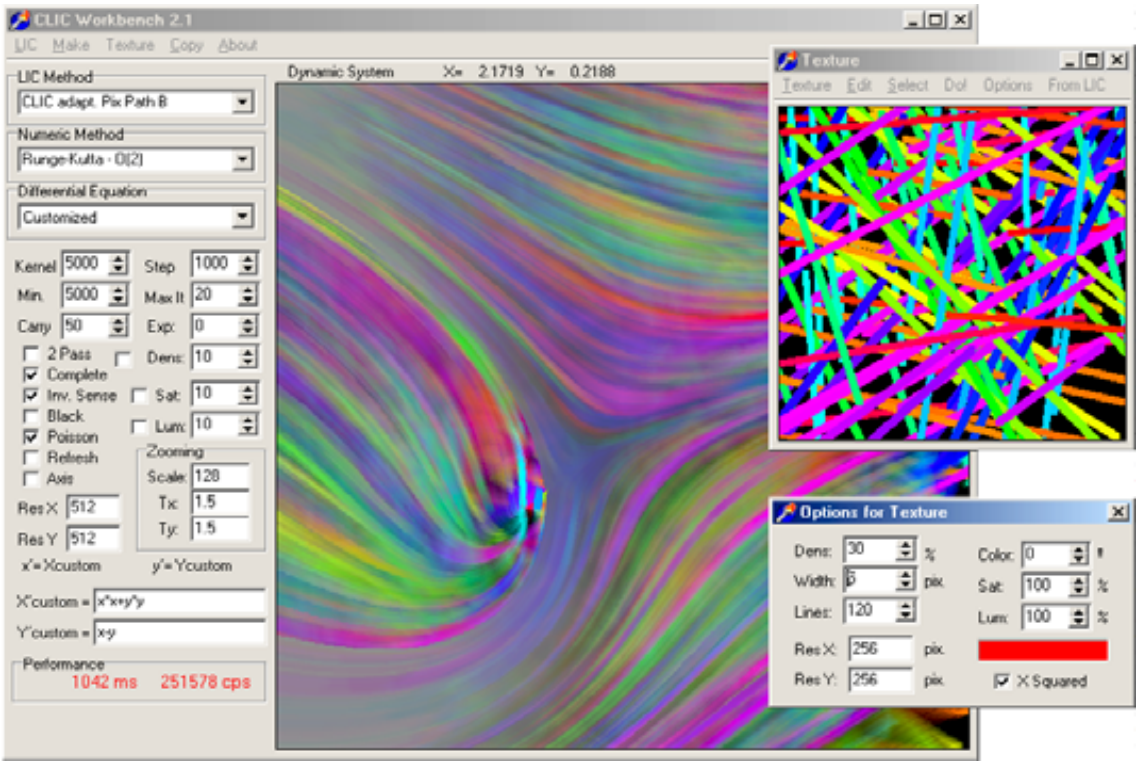



Figura 9: Interfase del CLIC Workbench.

# 5 Conclusiones y trabajo futuro

Se presentaron numerosas mejoras al algoritmo LIC para la visualización de diagramas de fases y flujos estacionarios. Los métodos propuestos combinan las mejores características de LIC y streamlines, es decir, calidad, autonomía y flexibilidad, con tiempos de cómputo muy bajos. Por dicha razón es que consideramos posible migrar esta aplicación a Java, de manera de poder proveer un servicio de visualización interactiva de sistemas dinámicos en Internet. También estamos trabajando en la implementación automática de animaciones basadas en cambios de la textura.

# Referencias

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

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

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

[4] Claudio Delrieux. Visualización de la Dinámica de Sistemas no Lineales. En *RPIC '97*, pages 151–157, San Juan, 1997.

[5] Claudio Delrieux. A Fast and Accurate Implementation of the Line Integral Convolution. En *Proceedings of the CISST 2000 Conference*, pages 467–474, CSREA Press, ISBN 1-892512-57-2, 2000.

[6] Claudio Delrieux, Julián Dominguez, y Andrés Repetto. Mejorando la Visualización de Campos Vectoriales. En *VI ICIE*, pages 397–416, International Conference on Information Engineering, 2001.

[7] Claudio Delrieux, Julián Dominguez, y Andrés Repetto. Streamlines y LIC: Visualización de Campos Vectoriales. En *WICC 2001*, páginas 193–196, Workshop de Investigadores en Ciencias de la Computación, 2001.

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

[9] Claudio Delrieux, Mirta Padín, y Gustavo Ramoscelli. Visualizing Bifurcations and Phase Portraits in Oscillating Systems. En *Proceedings of the CISST 2001 Conference*, páginas 687–694, CSREA Press, ISBN 1-892512-73-4, 2001.

[10] Claudio Delrieux y Andrés Repetto. Visualización de Sistemas no Lineales y Caóticos. En *Memorias XXV Conferencia Latinoamericana de Informática CLEI '99*, páginas 241–253, Asunción, Paraguay, 1999.

[11] Claudio Delrieux y Andrés Repetto. Métodos para la Visualización de la Dinámica de Sistemas no Lineales. En *Argentine Symposium on Computing Technology*, páginas 153–162, 29 JAIIO, 2000.

[12] Robert Devaney. *A First Course in Chaotic Dynamical Systems.* Addison Wesley,1992.

[13] L.K. Forssell y S.D. Cohen. Using Line Integral Convolution for Flow Visualization: Curvilinear Grids, Variable-speed Animation, and Unsteady Flows. *IEEE Transactions on Visualization and Computer Graphics*, 1(2):133–141, 1995.

[14] Hans-Christian Hege y Detlev Stalling. Fast LIC with Piecewise Polynomial Filter Kernels. En H.-C. Hege y K. Polthier, editores, *Mathematical Visualization*, páginas 295–314. Springer-Verlag, Amsterdam, 1998.

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

[16] M.H. Kiu y D.C. Banks. Multi-Frequency Noise for LIC. En *Visualization '96 Conference Proceedings*, páginas 121–126, Los Altos, CA, 1996. IEEE Press.

[17] D. Stalling y H.-C. Hege. Fast and Resolution Independent Line Integral Convolution. En *SIGGRAPH 95 Conference Proceedings*, páginas 249–256, Los Alamitos, CA, 1995.

[18] Stefen H. Strogatz. *Nonlinear Dynamics and Chaos.* Addison-Wesley, 1994.

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

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

[21] R. Wegenkittl, E. Gröller, y W. Purgathofer. Animating Flow-fields: Rendering of Oriented Line Integral Convolution. En *Computer Animation '97 Proceedings*, páginas 15–21, Los Alamitos, CA, 1997. IEEE Computer Society Press.