Visualisation de champs de vecteurs bidimensionnels à base de streamlines

Bruno Jobard

Thèse de Doctorat en Informatique

Soutenue le 10 janvier 2000
à l'Université du Littoral Côte d'Opale
devant la commission d'examen composée de :

Président	:	Edward Anthony, Professeur, ULCO
Directeur	:	Serge Petiton, Professeur, Université de Lille I
Co-Directeur	:	Wilfrid Lefer, Maître de Conférences, ULCO
Rapporteur	:	Bruno Arnaldi, Professeur, INSA de Rennes
Rapporteur	:	Gordon Erlebacher, Professeur, Université d'Etat de Floride
Rapporteur	:	Jean Favre, Docteur, Centre Suisse de Calcul Scientifique
Examinateur	:	Jean-Louis Pajon, Docteur, Institut Français du Pétrole

Remerciements

Je tiens sincèrement à remercier Wilfrid Lefer pour m'avoir offert l'occasion unique de faire une thèse, pour sa direction et son encadrement sur mesure et son soutien motivant lors des moments difficiles ou de découragement.

Je remercie Serge Petiton, directeur de thèse, Edward Anthony, président du jury ainsi que les membres du jury pour le temps qu'ils m'ont octroyé et particulièrement mes rapporteurs : Bruno Arnaldi, dont les discussions et conseils durant mon DEA à Rennes m'ont indiscutablement motivé à tenter l'aventure de la thèse ; Gordon Erlebacher pour son enthousiasme pour la recherche, pour m'avoir proposé un postdoctorat à l'université d'état de Floride (Florida State University) et me permettre de continuer mes travaux de thèse, et Jean Favre pour ses qualités humaines et son intérêt au sujet de mes recherches et de leurs potentialités pour l'avenir.

Je tiens à remercier les membres du Laboratoire d'Informatique du Littoral, conglomérat d'individualités passionnantes, avec qui j'ai vécu les premières années d'existence du LIL dans une ambiance chaleureuse et animée.

De mes quelques amis sur Calais, je tiens particulièrement à remercier les plus chers, Virginie et Philippe Marion, pour leur présence, leur art de vivre et leur hospitalité mainte fois sollicitée.

J'aimerais aussi mentionner Nicolas Beaudet, ami et associé exceptionnel des projets les plus fous, avec qui l'émulation durant le cursus universitaire nous a porté jusqu'à l'obtention de nos thèses respectives.

Ces remerciements ne seraient pas complets sans évoquer mes parents pour leur soutien inconditionnel et leur foi inébranlable dans la réussite de mes entreprises.

Table des matières

CHAPITRE

1

Introduction

1 Présentation du problème

La masse des données manipulées en science et en ingénierie tend à devenir gigantesque à mesure que les capacités de traitement des ordinateurs croissent. Ces données proviennent tant d'appareils de mesure tel que les satellites que de simulations de phénomènes physiques. Le but de la visualisation scientifique est de faciliter l'interprétation de ces masses de données en synthétisant des représentations visuelles intuitives pour un observateur humain. Les informations contenues dans ses bases de données sont de dimension variable, allant de données scalaires pour lesquelles on dispose d'une seule valeur en chaque point, aux données multi-variables composées d'une collection de valeurs. Quand les valeurs d'une donnée représentent une information directionnelle, cette donnée est alors appelée *vecteur*. Dans cette thèse, nous allons nous intéresser à la visualisation de *champs de vecteurs*.

Un champ de vecteurs peut être défini par un domaine spatial (2D ou 3D) sur lequel on dispose d'une information directionnelle en chacun de ses points. Les champs de vecteurs représentent un type d'information important en science puisqu'ils permettent de coder des phénomènes physiques tels que les fluides (courants en océanographie, vents en météorologie, ...), les champs électromagnétiques ou acoustiques, ou le comportement de systèmes dynamiques.

Depuis une dizaine d'années, de nombreuses techniques de visualisation ont été proposées afin de fournir des représentations adaptées aux informations véhiculées par les champs de vecteurs. L'objectif de ces techniques est principalement de représenter l'information directionnelle sur l'ensemble du domaine ainsi que la position de points particuliers, appelés «points critiques». De nombreuses méthodes ont été proposées afin de révéler la *structure* des champs de vecteurs, parmi lesquelles on peut distinguer celles produisant des représentations *éparses* ou *denses*. Une représentation est qualifiée d'éparse quand les caractéristiques du champ de vecteurs sont indiquées localement à l'aide d'objets graphiques distincts, des flèches par exemple. Par opposition, les

représentations denses sont obtenues par la synthèse de textures fournissant l'information souhaitée à l'échelle du pixel. Pour augmenter la richesse des représentations, quelques techniques ont été étendues afin de produire des animations révélant l'orientation des champs de vecteurs.

Tous nos travaux sont basés sur le placement et la coloration d'une courbe particulière qui peut être calculée en chacun des points du domaine couvert par le champ de vecteurs. Ces courbes sont les «lignes de champs», appelées «*field lines*» ou encore «*streamlines*». Nous montrons dans cette thèse qu'à partir de ces courbes, nous avons pu générer des représentations de champs de vecteurs 2D, allant de éparses à denses, par le placement de streamlines uniformément réparties, ainsi que des animations de champs stationnaires, puis non stationnaires. Notre approche cohérente nous a permis de proposer des méthodes compétitives vis à vis des techniques de visualisation existantes, tant en terme de temps de calcul que de rendu visuel.

2 Organisation de la thèse

Ce mémoire est organisé selon quatre principaux chapitres. Le premier présente un tour d'horizon des techniques utilisées en visualisation de champs de vecteurs tandis que les trois suivants exposent les travaux réalisés durant cette thèse.

Chapitre 2 : Techniques de visualisation de champs de vecteurs

Nous commençons ce chapitre par un bref exposé des concepts de base concernant les champs de vecteurs et les objectifs de leur visualisation. Les principales techniques de visualisation de champs de vecteurs disponibles dans la littérature sont ensuite présentées. Enfin nous précisons le mode de calcul des streamlines, qui sont à la base de nos travaux.

Chapitre 3 : Placement de streamlines

Une technique largement utilisée pour visualiser la structure globale d'un champ de vecteurs consiste à tracer un certain nombre de streamlines. Sans algorithme de placement particulier, les streamlines ont tendance à s'accumuler dans certaines zones de l'images en laissant d'autres parties désertes. Cette variation de contraste dans l'image perturbe l'interprétation de la structure du flux. Afin de supprimer ces artefacts visuels, nous avons développé un algorithme calculant un ensemble de streamlines uniformément réparties dans le domaine à visualiser. Des représentations de densité quelconque, allant de éparse à dense, sont obtenues en précisant la distance de séparation entre les streamlines. Nous verrons que notre algorithme s'avère beaucoup plus rapide que les méthodes proposées jusqu'alors.

Chapitre 4 : Animation de champs de vecteurs stationnaires

L'objectif est d'offrir une représentation intuitive de l'orientation en tout point du domaine grâce à une animation. Nous avons atteint ce but en animant des motifs de couleurs le long d'un unique ensemble dense de streamlines. Les animations produites sont parfaitement cycliques et permettent de visualiser les différentes vitesses locales du champ de vecteurs. Nous avons pour cela adapté notre algorithme de placement pour le calcul rapide d'un ensemble dense de streamlines et nous avons mis au point une méthode originale de corrélation des pixels qu'elles recouvrent. Une part importante de notre travail a été de stocker toutes les informations nécessaires à l'animation de N images dans une structure de données ayant la même taille qu'une image statique. Nous avons appelé cette structure originale : la *Motion Map*.

Chapitre 5 : Visualisation de champs non stationnaires

En animant des ensembles de streamlines calculés à des pas de temps successifs, il est possible de visualiser l'évolution de la structure d'un champ de vecteurs non stationnaire. Nous avons développé un mécanisme de corrélation, tant de la forme des streamlines que des motifs qui sont plaqués dessus, afin d'obtenir des animations fluides. Tout comme notre algorithme de placement pour les champs stationnaires, la densité de streamlines peut être simplement paramètrée afin de produire des représentations allant de éparse à dense. Cette densité est maintenue constante pendant toute l'animation. Le déplacement de textures mono-dimensionnelles le long des streamlines pendant la déformation de la structure du flux révèle son orientation.

En utilisant une approche uniforme, basée sur l'utilisation de streamlines, nous obtenons des résultats qualitativement comparables et généralement plus performants en temps de calcul en comparaison à ce qui a été précédemment proposé dans la littérature.

CHAPITRE

2

Techniques de visualisation de champs de vecteurs

1 Introduction

Nous commençons ce chapitre par quelques définitions relatives aux champs de vecteurs et nous exposons les objectifs de leur visualisation. Nous présentons ensuite l'état de l'art en visualisation de champs de vecteurs. Enfin, nous détaillons une méthode de calcul des streamlines, courbes issues des champs de vecteurs et fondement des techniques de visualisation que nous proposons dans cette thèse. Nous avons classé les différentes approches en quatre catégories :

- **Placement de primitives géométriques :** les informations à visualiser sont plaquées sur les attributs (forme, couleur, orientation...) de primitives graphiques réparties sur le domaine. Nous qualifions ces représentations d'«éparses» en raison de la faible densité d'objets dans l'image.
- **Génération de textures :** les textures sont des représentations particulièrement intéressantes pour la grande densité d'information qu'elles permettent de coder. Des efforts importants ont été effectués ces dernières années afin d'obtenir des représentations «denses».
- **Visualisation par l'animation :** un champ de vecteurs peut être vu comme le codage d'un mouvement. L'animation est alors un moyen naturel de visualiser son comportement dynamique. Il peut s'appliquer aussi bien aux représentations éparses que denses. La principale difficulté est de maintenir une bonne corrélation spatio-temporelle d'une image à l'autre de l'animation.
- **Extension aux flux non-stationnaires :** pour la plupart des algorithmes proposés pour la visualisation de champs de vecteurs stationnaires, des extensions ont été proposées pour le cas non-stationnaire. Ce cas reste à ce jour un problème difficile et les méthodes existantes ne sont pas encore tout à fait satisfaisantes.

2 Quelques définitions relatives aux champs de vecteurs

On définit un champ de vecteurs $\boldsymbol{f}$ pour toute valeur $\boldsymbol{x}$ d'un sous-ensemble E de l'espace Euclidien R^n dans l'espace vectoriel R^n comme une relation $\boldsymbol{f}(\boldsymbol{x}) : E \rightarrow R^n$. Cette fonction procure pour chaque point $\boldsymbol{x}$ de E une quantité vectorielle de dimension n, telle qu'une force ou une vitesse. Dans le cadre de cette thèse, nous nous restreindrons aux champs de vecteurs bidimensionnels, c'est à dire au cas où $n = 2$ et $E \subset R^2$. Si la quantité vectorielle est fixe au cours du temps, le champ de vecteurs est dit *stationnaire*, sinon il est dit *non stationnaire*, auquel cas ses valeurs dépendent du temps et on définit alors $\boldsymbol{f}$ tel que $\boldsymbol{f}(\boldsymbol{x}, t) : E \times J \rightarrow R^n$, où J est un intervalle de temps, $a \leq t \leq b$.

Suivant les domaines scientifiques ou techniques dont ils sont issus, les champs de vecteurs peuvent provenir de sources de natures différentes pouvant être le résultat de mesures expérimentales ou de simulations. Dans le premier cas, les valeurs sont mesurées en un certain nombre de points du domaine à l'aide de capteurs physiques. Dans le cadre de simulations, les champs de vecteurs sont obtenus en résolvant un système d'équations aux dérivées partielles modélisant le phénomène physique. Par exemple en mécanique des fluides, le comportement d'un fluide visqueux peut être modélisé par les équations de Navier-Stokes [9]. La forme générale du système pour un fluide incompressible non stationnaire est :

$$\begin{cases} \nabla . U = 0 \\ \dfrac{\delta U}{\delta t} + U . \nabla U + \nabla P - \upsilon \Delta U = 0 \end{cases} \tag{2.1}$$

où U est la vitesse du flux, t est le temps, P est la pression et υ est la viscosité.

Si la résolution analytique d'un tel système est possible, le champ de vecteurs est alors donné sous la forme d'une formule mathématique, sinon on a recours à des méthodes numériques afin de le résoudre [11]. Dans ce dernier cas, les vecteurs U sont calculés sur les noeuds d'une grille couvrant le domaine.

Pour avoir accès à une valeur en tout point du champ quand on ne dispose que de vecteurs en des positions discrètes, on a recours à une opération d'interpolation bi- ou trilinéaire des valeurs disponibles.

Quand le champ de vecteurs modélise le comportement d'un flux (les vecteurs sont les vitesses locales de ce flux), une distinction peut être faite entre les flux *laminaires* et *turbulents*. Dans les flux laminaires, tous les vecteurs dans un même voisinage sont à peu près les mêmes, contrairement au flux turbulents où de fortes variations se produisent dans une zone de petite taille.

Les caractéristiques que l'on souhaite observer dans les champs de vecteurs sont principalement la répartition des directions et des orientations des vecteurs sur le domaine ainsi que la position des *points critiques* (puits, sources, vortex...

- voir section 3.4). L'objectif de la visualisation de champs de vecteurs est de proposer des représentations (images ou animations) qui offrent une interprétation la plus aisée et intuitive possible des différents paramètres. La suite de ce chapitre est consacrée à la présentation des principales techniques de visualisation de champs de vecteurs proposées à ce jour dans la littérature.

3 Placement de primitives géométriques

3.1 Représentations icôniques

3.1.1 Flèches

Une représentation du champ de vecteurs peut être obtenue en dessinant des icônes (arrows, hedgehogs ou glyphs) en certains points du champ. Les hedgehogs sont de petits segments dont la longueur est proportionnelle à la vitesse et qui sont orientés selon la direction du flux. Les flèches apportent une information supplémentaire sur l'orientation. Cette technique est simple et efficace en terme de temps calcul. Toutefois, en 2D, il faut veiller à ce que les flèches ne se croisent pas pour garder une bonne lisibilité du flux ; cette contrainte limite la résolution spatiale. De plus, une distribution régulière des flèches peut introduire des artefacts visuels (création de motifs dans l'image) pouvant conduire à une mauvaise interprétation de la structure du champ. Pour pallier ce problème, Dovey [16] propose de distribuer les flèches en utilisant une méthode d'échantillonnage aléatoire du domaine.

En 3D, il est impossible de respecter cette contrainte car il faudrait tenir compte du point de vue afin d'éviter que les flèches ne se projettent en un amas dans certaines zones. De plus la position et la direction des vecteurs, une fois projetés dans le plan de l'image, est difficile à apprécier, en effet on ne sait pas si la flèche pointe vers l'avant ou vers l'arrière.

Pour améliorer la qualité de la visualisation, on peut utiliser des objets 3D à la place des flèches, ce qui permet de rendre l'effet de profondeur en utilisant un modèle d'éclairement tel que Phong. Malheureusement, la taille de ces objets est supérieure à de simples flèches. Il est alors nécessaire de réduire le nombre d'objets afin qu'ils ne se chevauchent pas dans l'image, réduisant ainsi la précision de la représentation. D'une manière générale, les flèches et les hedgehogs ne sont guère adaptés pour la représentation des champs de vecteurs 3D.

3.1.2 Icônes généralisées

Une simple flèche ne suffit plus quand des informations dérivées (courbure, vorticité, ...) doivent être visualisées en plus de la direction, de la vitesse et de l'orientation. Dans ce cas, des icônes plus complexes peuvent être utilisées.

Leur géométrie sera suffisamment riche pour représenter tous les attributs à visualiser. Ainsi, De Leeuw et van Wijk proposent une icône (appelée *flow probe*) pour la représentation locale de la dérivée spatiale du champ vitesse [12]. Le tenseur est alors défini en tout point de l'espace par une matrice 3×3. L'icône permet de visualiser à la fois le vecteur vitesse et les 9 valeurs du tenseur (voir figure 2.1).

A cause de leur taille et des problèmes d'occlusion dus à leur projection, les icônes ne peuvent pas être utilisées en nombre important dans une image.

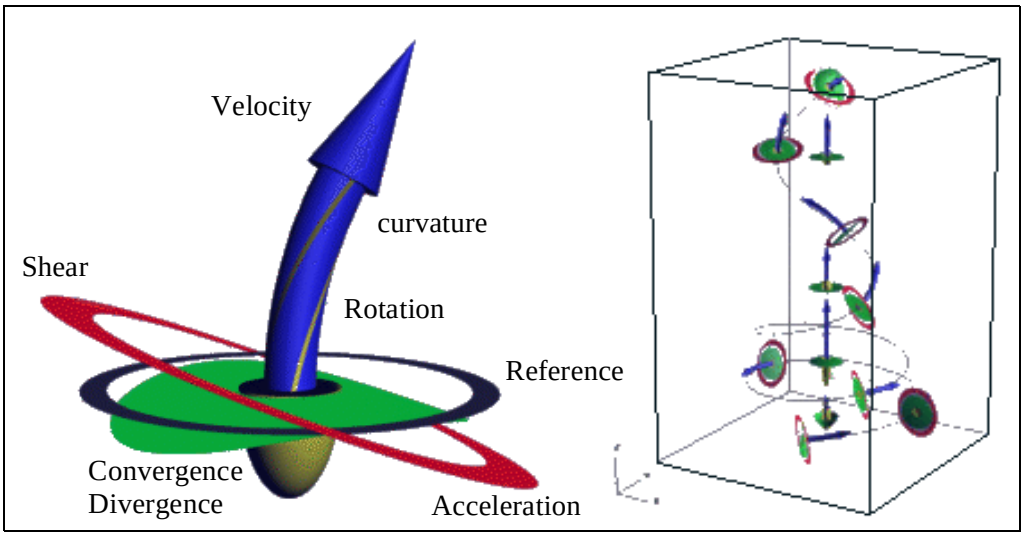

Figure 2.1 *Flow probe* : icône permettant de visualiser localement les caractéristiques d'un champ de tenseurs. (de Leeuw)

3.2 Visualisation par particules

L'utilisation de particules est très naturelle en visualisation de flux puisque directement calquée sur les procédés expérimentaux [47]. Les particules ne sont généralement pas ponctuelles mais ont la forme de petites ellipses dont l'axe principal est orienté selon la direction du flux. L'affichage des particules est souvent accompagné d'un effet de flou dû au mouvement (*motion blur)*, plus ou moins important suivant leur vitesse instantanée.

Stolk et van Wijk proposent d'utiliser des «particules surfaciques» [54][60]. Ces petites facettes, appelées *surface-particles*, possèdent une normale permettant de les orienter suivant des surfaces tangentes au flux. Les particules sont *semées* le long de rampes (lignes, cercles, etc...) et leurs positions sont déterminées de proche en proche par intégration. Si leur densité est suffisamment élevée, elles forment implicitement des (*stream*-) surfaces (voir section 3.3). Un accent particulier a été mis sur le rendu de ces particules pour en interpréter le plus facilement possible la position. Ainsi leurs ombres sont projetées sur des plans autour de la région d'intérêt pour en apprécier la

concentration (voir figure 2.2a). De plus un mécanisme de profondeur de champ (similaire à celui des appareils photographiques) à été ajouté pour focaliser l'attention de l'observateur sur une région particulière.

Max utilise aussi des particules surfaciques pour visualiser un flux 3D autour d'une surface [44]. Contrairement aux *surface-particles* de van Wijk, celles de Max peuvent se trouver à proximité de surfaces quelconques. En fait, les particules ne sont affichées que si leur distance à la surface d'intérêt est inférieure à une certaine valeur. Pour des raisons d'efficacité, le domaine 3D est divisé en cellules et des particules ne sont injectées que dans les cellules se trouvant à proximité de la surface (voir figure 2.2b). L'affichage peut être accéléré par l'utilisation d'un plaquage de texture câblé.

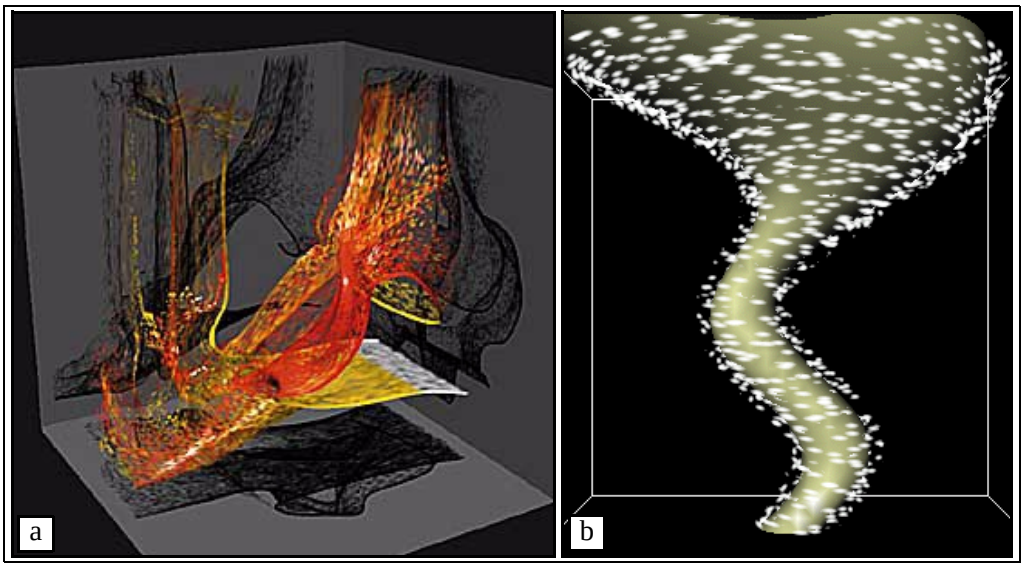

Figure 2.2 (a) *Surface-particles* de van Wijk et (b) particules autour d'une surface 3D de Max.

D'une manière générale, une vue statique, créée en utilisant des particules déformées dans le sens du flux, apporte une information qualitative sur la direction et la vitesse à l'emplacement de chaque particule, mais ne permet pas d'apprécier la structure globale du flux. Cette approche donne cependant des résultats intéressants quand les particules sont animées (voir section 5.1). Notons aussi que cette technique est encore très utilisée pour la visualisation de champs de vecteurs non-stationnaires pour lesquels l'éventail des méthodes efficaces disponibles n'est encore pas très étoffé (voir section 6.1).

3.3 Intégration de trajectoires et leur placement

Ces techniques, héritées de procédés expérimentaux, consistent à injecter dans le flux des marqueurs supposés sans masse en certains points du champ et de

visualiser leur trajectoire. Ces marqueurs peuvent donner naissance à plusieurs variétés de courbes, que nous présentons maintenant.

- Une ***streamline*** est une courbe tangente à la vitesse du flux en tout point. Les streamlines permettent de représenter la structure instantanée d'un champ de vecteurs.
- Une ***pathline*** est la trajectoire d'une particule individuelle se déplaçant dans le flux. Une telle courbe correspond aux positions successives de la particule dans le temps.
- Une ***streakline*** est la courbe qui relie des particules lâchées en flot continu à partir d'une source ponctuelle immobile. Toutes les particules dérivant dans le flux au cours du temps, la structure de la courbe évolue d'un instant à l'autre.
- Une ***timeline*** est la courbe qui relie plusieurs particules injectées au même instant à partir d'un ensemble de points appartenant à une droite. La déformation de cette courbe au cours du temps donne une information sur la structure du flux.

Quand le flux est stationnaire, les streamlines, pathlines et streaklines se confondent et le terme de streamline est alors préféré. Les pathlines et streaklines sont utiles dans le cas de flux non-stationnaires, c'est à dire quand la structure du champ de vecteurs évolue au cours du temps.

Bien que ces différentes courbes soient des outils particulièrement utilisés pour les flux bi-dimensionnels, leur interprétation en 3D est souvent ambiguë du fait du problème de la perception des lignes en 3D. Les images deviennent alors très vite confuses. Les problèmes d'interprétation proviennent de la difficulté à pouvoir évaluer la courbure et la position de la courbe dans le domaine. Il est toutefois à noter que des modèles d'illumination pour les objets 1D plongés dans l'espace 3D ont été proposés (Banks [2], Stalling [53], Zöckler [62]) (voir figure 2.3).

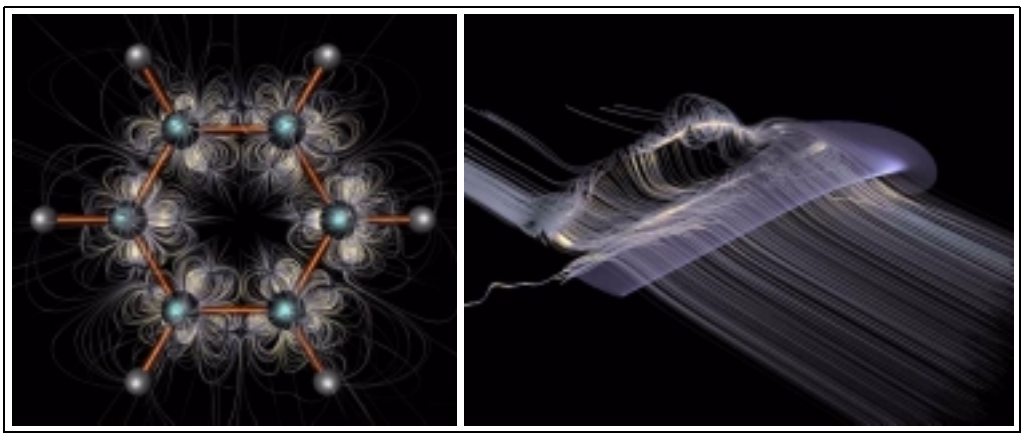

Figure 2.3 Streamlines illuminées pour visualiser leurs courbures (par Zöckler et Stalling)

En 3D, ces courbes mono-dimensionnelles peuvent être utilisées afin de construire des primitives graphiques plus complexes, de visualisation moins ambiguë et apportant plus d'informations sur les caractéristiques du flux. Voici les principales primitives ayant été proposées dans la littérature :

- Les ***streamballs*** sont des sphères centrées sur les points d'une trajectoire. L'espacement entre chaque paire de points consécutifs est proportionnel au vecteur vitesse. Ainsi dans les zones de grande vitesse les sphères sont distinctes alors qu'elles se confondent partiellement dans les régions de faible vitesse (formant des tubes ondulés) (Brill [4]) (voir figure 2.4a).
- Un ***streamribbon*** est un ruban dont un bord est une streamline et l'autre est une ligne brisée reliant les extrémités de vecteurs perpendiculaires à cette streamline. Les vecteurs perpendiculaires sont obtenus successivement à chaque pas d'intégration par rotation d'un vecteur initial (Ueng [56]). En plus de la structure du champ de vecteurs tridimensionnel, le streamribbon permet de visualiser la rotation locale du flux.
- Un ***streamtube*** est construit en reliant des cercles centrés sur un point d'une streamline et orientés perpendiculairement à cette streamline (Ueng [56]). Un attribut (vitesse locale, pression, etc) peut être utilisé pour contrôler le rayon du tube le long de la streamline (voir figure 2.4b). Plus généralement, le cercle peut être remplacé par tout autre polygone. Il s'agit alors de ***streampolygons*** (Shroeder [50]). La forme n'étant plus symétrique, par rotation autour de la streamline, elle peut être utilisée afin de visualiser la rotation locale du flux.
- Une ***streamsurface*** est une surface tangente au vecteur vitesse du champ en tout point. Elle est construite en reliant transversalement un certain nombre de streamlines par des polygones (Hultquist [29]). La difficulté est de maintenir un maillage relativement uniforme entre ces streamlines qui ont tendance à s'éloigner ou se rapprocher les unes des autres. Van Wijk donne une construction alternative en considérant la streamsurface comme une isosurface d'une fonction scalaire (van Wijk [61]). Pour réduire le problème d'occlusion qui se produit quand plusieurs streamsurfaces sont visualisées, Löffelman propose un technique appelée *streamarrows* [39]. Elle consiste à plaquer sur les streamsurfaces des textures ayant des régions transparentes. Ces régions transparentes ont une forme de flèches afin d'indiquer l'orientation du flux.
- Un ***flow volume*** est construit en laissant dériver un polygone générateur dans le flux. A chaque pas de temps un nouveau polygone est obtenu en intégrant les sommets du polygone précédent. Un maillage relie ces différents polygones. Comme pour les streamsurfaces, il faut s'assurer que le maillage ne dégénère pas. Pour cela il est souvent nécessaire de subdiviser le polygone générateur pour tenir compte de la divergence du flux. L'intérêt de cette approche est l'aspect volumique de l'objet créé. Il permet notamment d'associer aux tétraèdres le constituant, une opacité fonction de leur volume. Un résultat possible est la réalisation de fumées (Max [43], Ma [40]) dérivant au gré du flux (voir figure 2.4c).

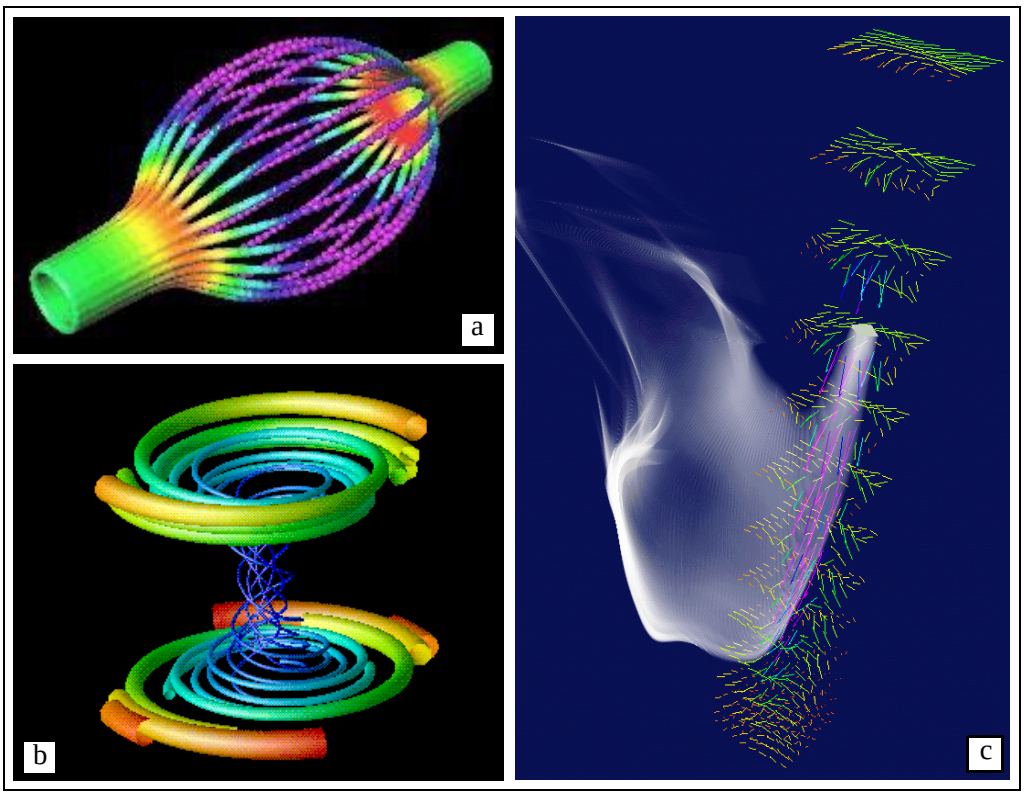

Figure 2.4 Exemples de visualisation obtenues par (a) *streamballs*, (b) *streamtubes* et (c) *flow volumes*.

Pour toutes ces techniques de visualisation s'appuyant sur des trajectoires, le placement des primitives dans le domaine a une influence très importante sur la qualité de la visualisation. Généralement, plus la taille des primitives utilisées est grande et moins elles seront nombreuses dans une image, à cause des problèmes d'occlusion. Dans ce cas, l'information apportée par la visualisation ne repose que sur quelques trajectoires. Leur placement doit donc être judicieux afin de révéler le plus efficacement possible les caractéristiques du flux. Lorsque le nombre de primitives est très faible, il pourra être effectué empiriquement par l'utilisateur, ou interactivement si la technique le permet, jusqu'à atteindre un placement satisfaisant. Lorsque le nombre de trajectoires à visualiser est plus important, il convient d'utiliser un algorithme de placement.

Les méthodes les plus simples consistent à initier ces courbes de manière aléatoire ou a partir des noeuds d'une grille couvrant le domaine. Ces méthodes simples sont toutefois rarement satisfaisantes au niveau du rendu visuel (accumulation de trajectoires dans certaines régions, déficit dans d'autres). Un placement optimum doit garantir la couverture du domaine à visualiser avec un ensemble minimal de trajectoires. Ainsi on améliore la qualité de l'image et on diminue le temps de calcul, en minimisant le nombre d'intégrations numériques.

Turk et Banks ont proposé un algorithme itératif de placement de streamlines [54]. Le domaine est d'abord couvert par un ensemble de streamlines placées aléatoirement puis l'image est améliorée par raffinements successifs. Les modifications autorisées sur cet ensemble de streamlines sont des opérations de déplacement, allongement, connexion, création et destruction. Une énergie calculée sur une version filtrée de l'image obtenue sert de critère d'optimisation. L'intérêt de cette méthode est le placement harmonieux de streamlines relativement longues, imitant le style des illustrations à main levée. Son inconvénient majeur est la lenteur du processus d'optimisation qui modifie chaque streamline aléatoirement et ne valide le changement que si l'énergie globale de l'image se trouve diminuée. Il est aussi à noter que l'algorithme ne termine pas seul et nécessite l'intervention d'un utilisateur pour stopper l'optimisation à un stade jugé suffisant (pas de critère d'évaluation globale de la qualité de l'image obtenue). Cette méthode de placement a été récemment étendue au placement de streamlines sur des grilles curvilinéaires (Mao [42]).

Le chapitre 3 de cette thèse est consacré à l'algorithme de placement de streamlines que nous avons développé [30]. Une caractéristique importante est qu'il dispose toutes les streamlines à leur emplacement final en une seule passe. Un seul paramètre, la distance de séparation entre les streamlines, suffit à spécifier leur densité dans le domaine. Notre algorithme est très efficace en comparaison de ceux existant pour des résultats visuels similaires. Signalons que depuis sa publication, notre travail a été étendu par Fuhrmann et Gröller au placement des streamlines en 3D (Fuhrmann [20]).

3.4 Représentation de la topologie du champ de vecteurs

L'idée est d'épurer la représentation du champ de vecteur afin de n'en retenir que sa topologie et ainsi éliminer les informations redondantes. On visualise la topologie d'un champ de vecteurs en construisant des courbes (ou des surfaces en 3D) reliant les points critiques entre eux (points où le vecteur vitesse est nul) (voir figure 2.5).

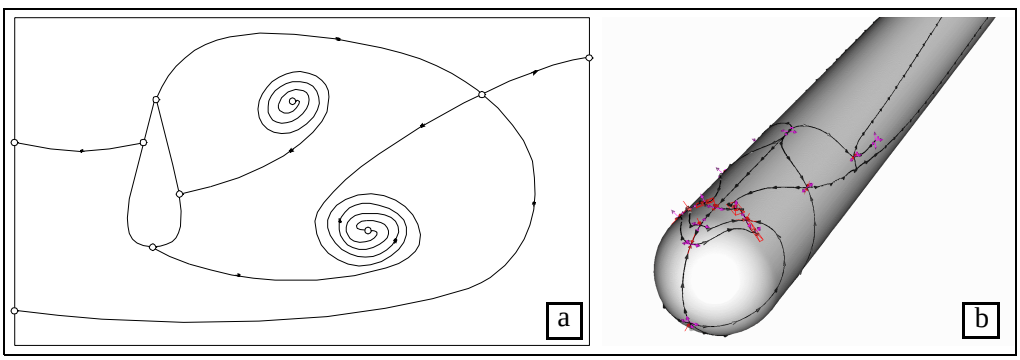

Figure 2.5 **Topologie de champs de vecteurs**. (a) en 2D et (b) à la surface d'un objet 3D.

Helman et Hesselink [25][26][27] ont proposé un système pour visualiser la topologie des flux. Globus et al. donnent des informations détaillées sur l'implémentation d'un module de visualisation de la topologie des champs de vecteurs [20], où la visualisation est enrichie par l'utilisation d'icônes précisant la nature des points critiques.

Le processus de visualisation de la topologie d'un champ de vecteurs procède en trois étapes :

- Calcul de la position des points critiques (vecteur vitesse nul). Ceci est effectué en scrutant le flux à la recherche des passages par zéro. Cette étape n'est pas triviale car généralement peu de vecteurs dans la représentation du champ sont nuls. La position des points en lesquels la vitesse s'annule doit être trouvée par interpolation ou par des méthodes numériques.
- Classification des points critiques en examinant les vitesses au voisinage de ces points, par un développement de Taylor par exemple. On distingue ainsi les points répulsifs, attractifs, en spirales, en ellipses, etc.
- Construction de courbes tangentes aux flux reliant les points critiques les uns aux autres. Dans le cas 3D, les points et courbes ne suffisent plus, les courbes servent alors à la construction de surfaces tangentes aux flux (*streamsurfaces*) qui relient ces différentes courbes critiques.

Cette technique permet une vue d'ensemble du champ de vecteurs mais donne une information locale très limitée. De plus, l'analyse de la topologie devient difficile quand le flux est très turbulent (quand les points critiques sont proches les uns des autres).

4 Génération de textures

Une limitation avec l'utilisation de primitives graphiques telles que des flèches ou des objets géométriques, est la faible résolution spatiale qu'elles permettent. Si on accroît le nombre d'objets, des problèmes d'occlusion surviennent, rendant l'interprétation de l'image difficile. Une alternative est donc l'utilisation de méthodes qui génèrent des textures afin de donner une information directionnelle dense sur tout le champ de vecteurs. Avec de telles techniques, la résolution spatiale est maximale puisqu'elle ne dépend que de la taille d'un pixel. De plus, les textures sont des objets très facilement intégrables dans des environnement 3D puisqu'il est relativement aisé de les plaquer à la surface de n'importe quel objet. Enfin, les cartes graphiques avec plaquage de texture câblé sont maintenant courantes et bon marché, ce qui autorise le rendu en temps réel de telles images.

4.1 Codage par disque de couleurs

Cette technique consiste à créer une texture dont la couleur de chaque texel (élément de texture) est déterminée en fonction de certains attributs du champ de vecteurs au point considéré (Johannsen [32]). L'algorithme travaille en HLS (chaque couleur est définie par sa teinte, sa luminance et sa saturation). L'espace des couleurs utilisé est le cercle S=1.0, L=0.5, H étant déterminé en fonction de la direction du vecteur vitesse. On peut également fixer L ou S en fonction de la norme du vecteur vitesse. Une telle représentation est très rapide à calculer et possède une résolution spatiale optimale (à l'échelle du pixel). Le disque des couleurs est représenté à proximité du champ visualisé, faisant ainsi office de légende au dessin. Ce codage des directions du flux par des couleurs n'est toutefois pas très intuitif et ne permet pas une interprétation aisée de la structure du flux.

4.2 Spot noise

Le principe de cette méthode, proposée par van Wijk [58], est de créer une texture en projetant des primitives spatiales élémentaires, appelées *spots*, réparties aléatoirement dans l'image. Les primitives sont de petits disques qui sont déformés à surface constante selon un axe principal de déformation qui est en fait la direction du flux au point de projection (voir figure 2.6). Les formes 2D obtenues sont donc des ellipses. Dans la version initiale, la rigidité des spots était mal adaptée à la représentation de flux comportant des courbures importantes. Une extension de la méthode (De Leeuw [12]) a donc consisté à courber l'axe principal des spots selon une streamline de façon à suivre les changement de direction du flux sur toute la longueur du spot. Pour visualiser l'orientation du flux, il est possible d'animer la texture en intégrant la position de chaque spot d'une image à l'autre (voir section 5.4).

La qualité des textures obtenues dépend fortement du nombre de spots utilisés. Un compromis doit donc être fait entre qualité de la texture et temps de calcul.

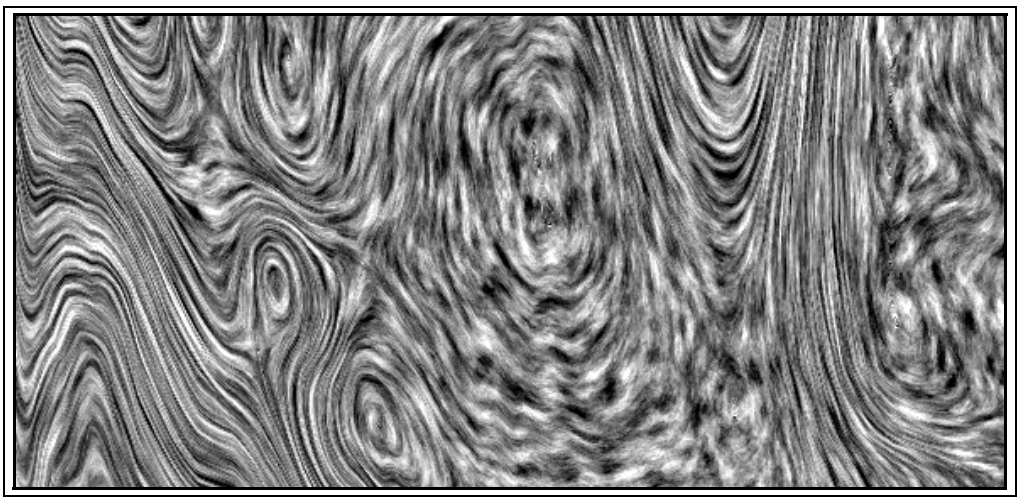

Figure 2.6 Texture Spot Noise.

Un aspect intéressant de cette méthode est l'utilisation de primitives graphiques, que sont les spots, pour la création des textures. Le rendu final peut être influencé par le choix de leur position. Ainsi, il est possible, par exemple, d'obtenir des images semblables à celles obtenues par les techniques expérimentales à base de particules d'huile ; l'accumulation de l'huile dans certaines régions de la surface permet de montrer la convergence du flux. Cet effet est obtenu avec spot noise en faisant dériver, dans une phase d'initialisation, les positions aléatoires des spots au gré du flux. C'est seulement après quelques pas d'intégration que les spots sont projetés dans la texture finale. Les spots se trouvent alors concentrés dans les zones de convergence (voir figure 2.7).

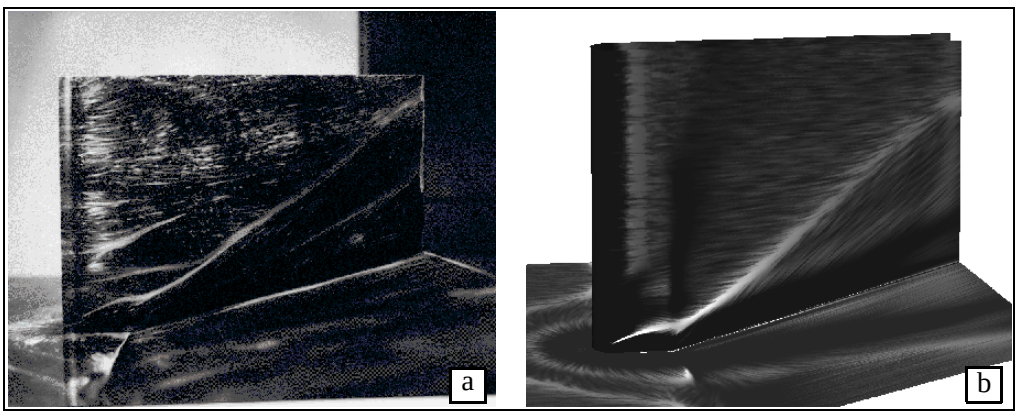

Figure 2.7 Visualisation expérimentale par projection de particules d'huile sur une surface (a), simulation de ce procédé par Spot Noise (b).

4.3 Line Integral Convolution (LIC)

Cette technique a été à l'origine de nombreux travaux depuis 6 ans. Elle a été introduite par Cabral et Leedom [6] en 1993 puis améliorée et étendue par la suite.

La *convolution* est une opération courante en traitement d'image. Etant donné une image d'entrée, chaque valeur des pixels de l'image de sortie est calculée comme une moyenne pondérée des valeurs d'une petite région de pixels de l'image d'entrée. La pondération utilisée pour multiplier les valeurs des pixels d'entrée est définie par une fonction appelée *noyau de convolution*. Suivant la dimension de la région que couvre le noyau de convolution dans l'image d'entrée, on peut les classer comme des convolutions uni-, bi- ou tri-dimensionnelles. La convolution est souvent utilisée pour lisser (*smoothing*) une image ou calculer des dérivées. Les paragraphes suivants montrent comment la convolution est utilisée dans l'algorithme LIC.

Prenons l'exemple simple d'un noyau de convolution uni-dimensionnel supporté par un segment de droite horizontal. La convolution de l'image 2.8a dont les pixels ont des valeurs aléatoires (*bruit blanc*) donnera l'image de même taille 2.8b. Pour cela chaque pixel de l'image de sortie est obtenu en moyennant le pixel correspondant avec les *n* pixels (*n* est fixé) à sa gauche et à sa droite dans l'image d'entrée. Cet exemple de convolution uni-dimensionnelle peut être étendu à la création de textures avec des lignes ou courbes ayant des orientations arbitraires.

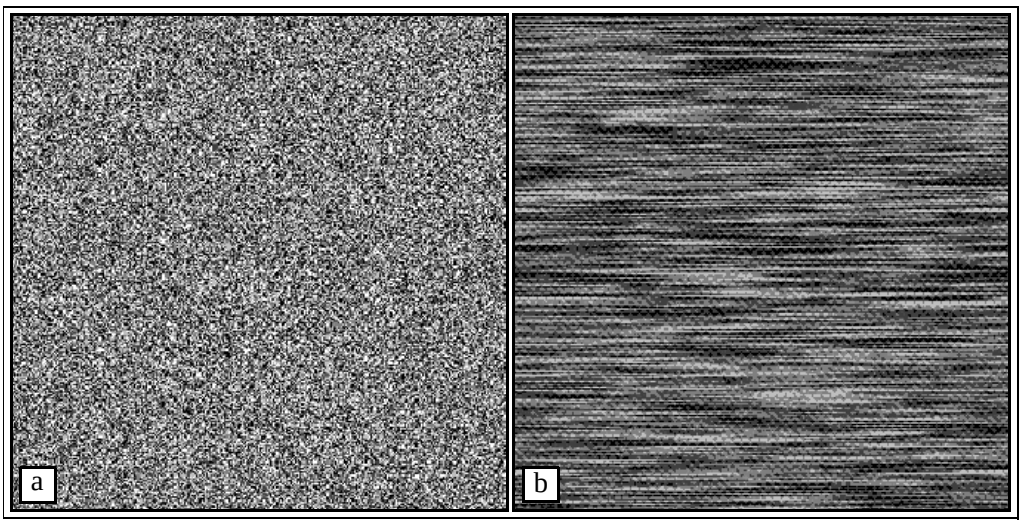

Figure 2.8 **Convolution uni-dimensionnelle.** (a) Image d'entrée (bruit blanc), (b) image de sortie dans laquelle les lignes du noyau de convolution apparaissent

Le principe de LIC consiste à choisir une streamline passant par le pixel à calculer comme support du noyau de convolution. L'algorithme prend en entrée un champ de vecteurs défini sur une grille cartésienne et une image d'entrée (généralement un bruit blanc). La streamline support est intégrée dans les deux directions à partir de la position du pixel courant dans le champ de vecteurs. La valeur des pixels d'entrée se trouvant sur la trajectoire de cette streamline est multipliée au cours de l'intégration par un coefficient de pondération calculé comme l'intégrale exacte de la fonction *k* du noyau de convolution. Ces coefficients de pondération peuvent être exprimés de la façon suivante :

$$h_i = \int_{s_i}^{s_i + \Delta s_i} k(w)dw$$

où Δs_i est la taille (longueur) du $i^{ème}$ pas d'intégration et s_i est la distance que la streamline a parcouru après *i* pas.

$$s_0 = 0 \quad \text{et} \quad s_i = s_{i-1} + \Delta s_{i-1}$$

En utilisant le résultat de l'intégration h_i comme variable de pondération, on peut calculer le résultat de la convolution pour chaque pixel comme :

$$C_{out}(x,y) = \frac{\sum_{i=0}^{l} C_{in}(P_i)h_i + \sum_{i=0}^{l'} C_{in}(P'_i)h'_i}{\sum_{i=0}^{l} h_i + \sum_{i=0}^{l'} h'_i}$$

où $C_{out}(x,y)$ est la valeur du pixel de sortie en (x,y), $C_{in}(P_i)$ est la valeur du pixel du bruit blanc d'entrée au point P_i, P_i est la position au $i^{ème}$ pas d'intégration de la streamline dans la direction positive, P'_i représente le pas correspondant dans la direction négative, $P_0 = (x,y)$, l et l' sont les distances d'intégration suivant les direction positive et négative respectivement, et h_i et h'_i sont les variables de pondération décrites ci-dessus. La figure 2.9 illustre le processus de convolution pour un pixel.

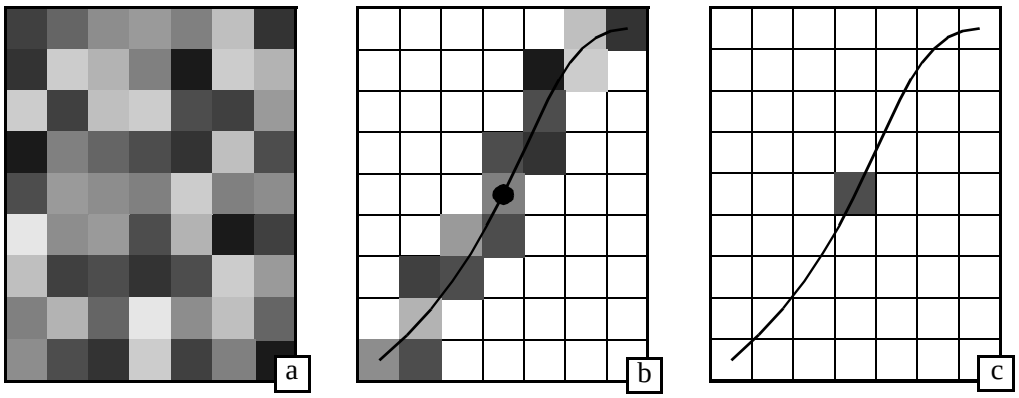

Figure 2.9 **Processus de convolution pour LIC**. Pour calculer un pixel de la texture finale (c), on intègre une streamline de part et d'autre du pixel (b) et on moyenne les intensités des pixels traversés dans la texture d'entrée.

Dans l'expression précédente de la convolution, l'utilisateur ne doit préciser qu'un paramètre, qui est la longueur du noyau de la convolution *L*, égale à la somme des variables *l* et *l'*. Quand *L* augmente, le rendu de la convolution est plus lisse et les lignes de champ apparaissent plus continues mais le calcul est aussi plus coûteux. La figure 2.10 montre une image obtenue avec l'algorithme LIC à partir d'un bruit blanc pour des valeurs de *L* différentes.

L'algorithme LIC peut facilement être étendu à la visualisation de champs de vecteurs 3D en utilisant en entrée une texture 3D. Les streamlines sont intégrées dans le champ de vecteurs 3D et le résultat de la convolution est stocké sous forme d'un volume de scalaires. Un algorithme de visualisation volumique directe peut être utilisé pour visualiser un tel volume en utilisant des fonctions de transfert indexées sur la vitesse, la pression ou la température du fluide par exemple.

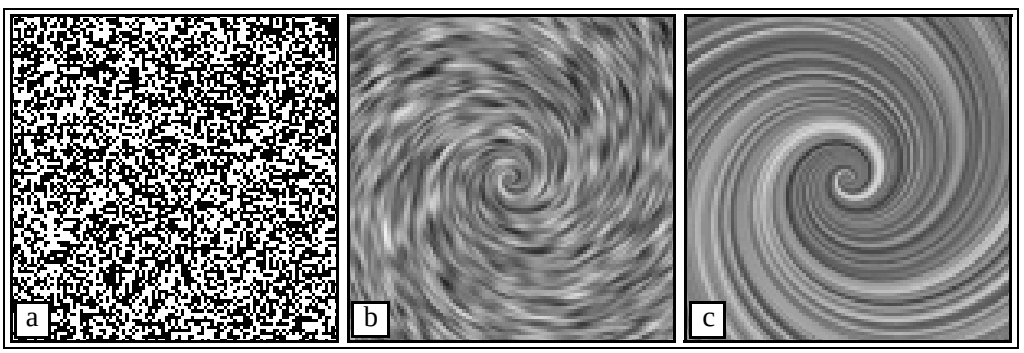

Figure 2.10 Visualisation d'un champ de vecteur 2D en utilisant l'algorithme LIC. Un bruit blanc (a) est convolué avec des valeurs de *L* différentes (b et c).

Dans sa version initiale, l'algorithme LIC ne fonctionne qu'avec un champ de vecteurs défini sur une grille cartésienne. Lisa Forsell a proposé de l'étendre aux grilles curvilinéaires, courantes en sortie des systèmes de simulation en dynamique des fluides [17][18]. Cette extension consiste principalement en l'ajout d'une phase préalable de conversion des données définies dans l'espace physique (grille curviligne) vers un espace de calcul plus propice : une grille cartésienne. Une fois la texture calculée, elle est plaquée sur le modèle physique à l'aide des techniques traditionnelles de plaquage de textures (voir figure 2.11). Il est toutefois recommandé, quand les grilles curvilignes sont irrégulières, de les reéchantillonner de façon à minimiser les déformations de la texture lors du plaquage (Okada [46]). L'algorithme LIC a aussi été étendu aux surfaces quelconques définies par un maillage triangulaire (Mao [41], Battke [3]). Dans ce dernier cas, le champ de vecteurs est tri-dimensionnel et sa visualisation est faite à la surface d'objets 3D plongés dans le même domaine.

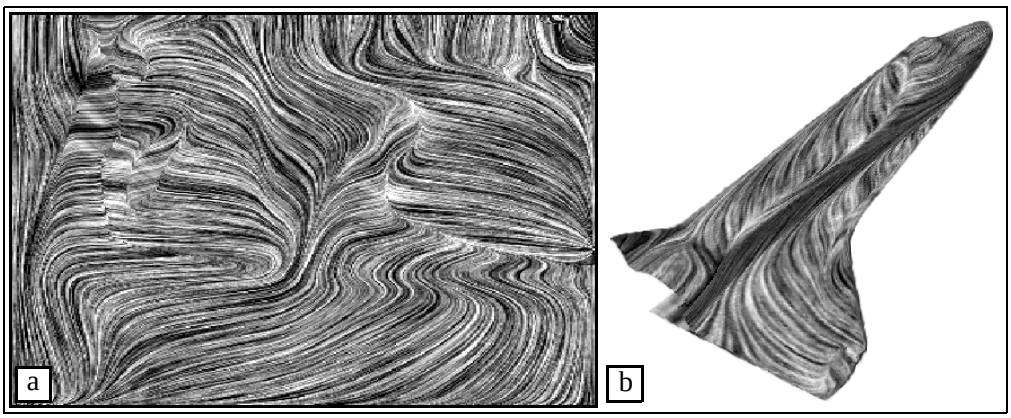

Figure 2.11 Texture LIC convoluée dans l'espace de calcul puis plaquée dans l'espace physique.

fastLIC, présenté par Stalling et Hege [52], est une variante accélérée de LIC. Leur travail est basé sur deux observations principales. La première est qu'une streamline initiée en un point du domaine traverse à elle seule beaucoup de pixels. Les pixels peuvent ainsi partager cette streamline lors de la convolution de façon à éviter des pas d'intégration numérique redondants (l'intégration des streamlines est coûteuse). Leur seconde observation est que des pixels voisins sur une même streamline utilisent un ensemble de pixels similaire pour la convolution. Ainsi la valeur LIC d'un pixel peut être réutilisée par ses voisins en la modifiant légèrement afin d'accélérer la convolution. Grâce à la réduction du nombre de streamlines calculées et l'accélération de la convolution, des gains conséquents ont été réalisés (de l'ordre d'un facteur 10). Ils s'affranchissent aussi de la contrainte de résolution constante entre le champ de vecteurs et les textures d'entrée et de sortie qui existe dans l'algorithme

original. Cette indépendance des résolutions permet de zoomer à volonté dans le domaine à visualiser et en facilite ainsi son exploration.

La qualité visuelle des images LIC est cependant pénalisée par leur faible contraste. Cet inconvénient est inhérent à l'opération de convolution elle-même. Le moyennage des valeurs des pixels du bruit blanc d'entrée a tendance à concentrer les valeurs des pixels de sortie sur une plage étroite d'intensités. Ainsi Okada conseille le passage de filtres pour augmenter le contraste de l'image de sortie [46]. Il est aussi possible d'appuyer le dessin des lignes de courant dans l'image en exécutant plusieurs fois l'algorithme avec en entrée la texture obtenue en sortie de l'exécution précédente (généralement deux passes suffisent).

Pour visualiser l'intensité de la vitesse en plus de la direction, Kiu et Banks ont eu l'idée de fournir en entrée de l'algorithme de LIC un bruit blanc dont la fréquence spatiale est fonction de l'intensité du flux [33]. Une série de textures de référence de fréquences différentes est créée. Leur fréquence représente une plage des vitesses du flux (les basses fréquences pour les vitesses les plus rapides). Le bruit blanc multi-fréquence est construit en choisissant ses pixels dans la texture dont la fréquence correspond à l'intensité de la vitesse locale du flux. En ne jouant que sur la texture d'entrée, cette approche permet à l'algorithme LIC (inchangé) de visualiser en plus de sa direction, la vitesse locale du flux (voir figure 2.12).

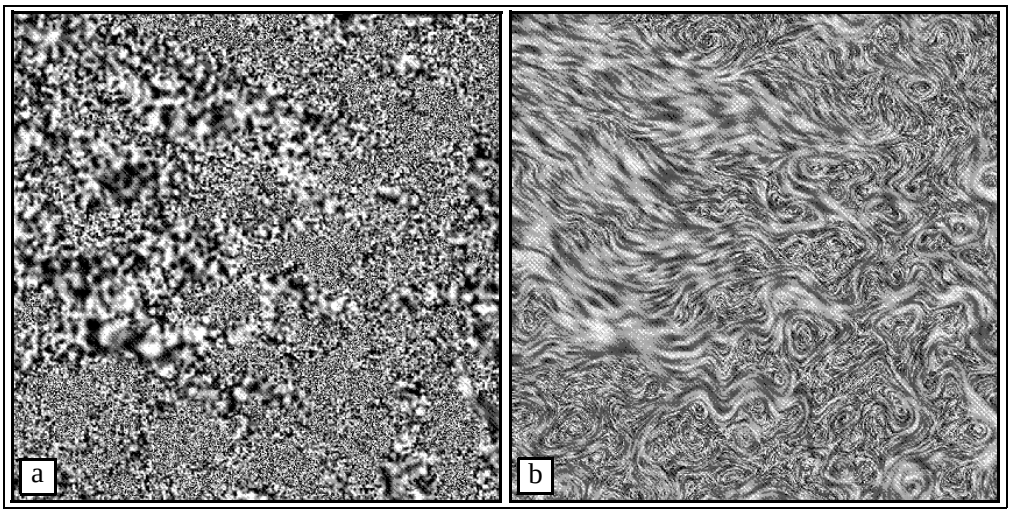

Figure 2.12 Visualisation de l'intensité de la vitesse avec LIC. En utilisant une texture multi-fréquence (a) en entrée de l'algorithme, les vitesses plus élevées apparaissent dans la texture LIC dans les zones de basses fréquences (b).

A ce stade, les informations véhiculées par une image LIC sont la direction du flux et sa vitesse. L'orientation du flux n'est pas directement interprétable sur une image fixe mais peut être visualisée par une animation (voir section 5.5).

Une alternative à l'animation pour la visualisation de l'orientation du flux à été proposée par Wegenkittl [57]. Il s'agit de l'algorithme OLIC pour *Oriented Line Integral Convolution*. Son principe repose sur l'utilisation d'un noyau de convolution asymétrique. Une version accélérée de leur algorithme (FROLIC) est présentée dans [58].

5 Visualisation par l'animation

Il est intéressant de rendre la dynamique des champs de vecteurs codant un mouvement par une animation. Les techniques précédentes permettent de visualiser un champ de vecteurs sous forme d'une image statique. Plusieurs d'entre elles ont été étendues afin de produire des animations.

5.1 Animation de particules

La visualisation d'un champ de vecteurs en animant des particules qui semblent portées par le flux est très naturelle. Elle est l'équivalent des lâchés de particules effectués en mécanique des fluides expérimentale. La mise en oeuvre se résume à intégrer la position des particules d'un pas de temps au suivant. Cette technique permet une interprétation assez intuitive de la direction, de l'orientation et de la vitesse. La partie délicate est la gestion de la répartition des particules dans le domaine à visualiser. En effet, les particules ont tendance à se concentrer dans certaines régions du domaine, rendant impossible l'interprétation du flux dans les zones désertées. Un mécanisme d'injection de particules doit être alors entretenu dans les zones de faible densité, tout en minimisant les artefacts visuels dus à l'apparition de nouvelles particules.

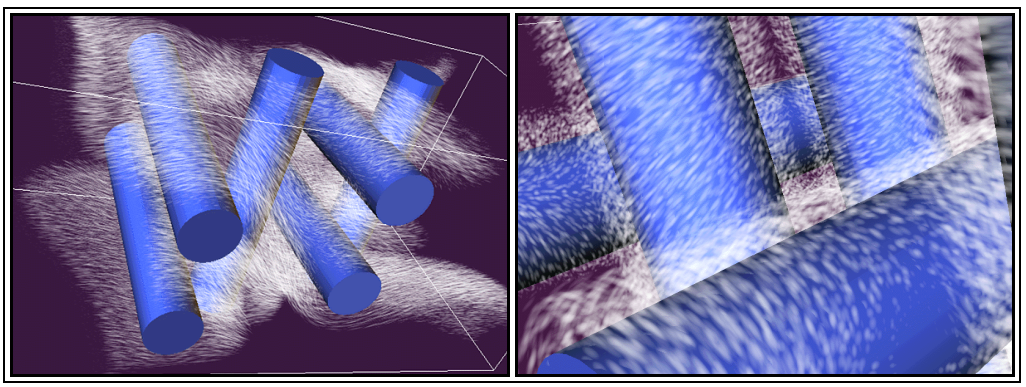

Figure 2.13 Animation de particules entre des cylindres

5.2 Animation de la palette des couleurs

Van Gelder utilise la technique d'animation de la palette des couleurs présentée par Shoup en 1979 [50] pour visualiser en temps réel l'animation de flux 3D [21]. Son approche consiste à intégrer des streamlines représentées par des tubes dont les bords sont noirs et le coeur est constitué d'un agencement particulier des couleurs d'une palette. Quand une rotation des indices des couleurs est faite dans la palette, le flux semble se déplacer à l'intérieur de ces tubes. Les tubes sont initiés à l'intérieur d'un volume spécifié par l'utilisateur et sont rendus à l'aide d'un Z-buffer. A cause de l'occlusion due à l'épaisseur des tubes, seule l'enveloppe extérieure de la région d'intérêt est en pratique visualisée (voir figure 2.14).

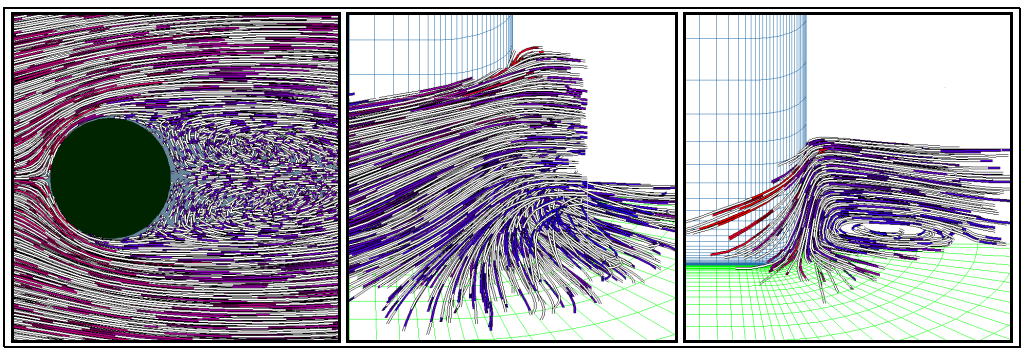

Figure 2.14 En animant la palette des couleurs de ces images, le flux semble se déplacer à l'intérieur des tubes.

5.3 Déformation de textures

Cette méthode consiste à déformer au cours du temps une texture de référence dans la direction des vecteurs du champ auquel elle est superposée (Max [45]). Le domaine à visualiser est partitionné en un maillage triangulaire régulier, fixé pour toute l'animation. A l'initialisation, les coordonnées de texture affectées aux sommets des triangles correspondent à leur position relative dans le domaine. Une technique de plaquage de texture est utilisée afin de gérer les déformations (étirement, compression et rotation) à apporter aux triangles de texture de référence de façon à couvrir tous les pixels des triangles de l'image finale. Pour un sommet donné, ses nouvelles coordonnées de texture dans l'image suivante sont obtenues en intégrant la position actuelle dans la texture selon l'opposé du vecteur en ce point. C'est le changement successif des coordonnées de textures au sommet de ces triangles qui donnera l'effet de déformation le long du flux.

Après plusieurs pas de temps, la déformation de la texture de référence est trop importante et la structure de la texture devient imperceptible. Une première

solution consiste à réinitialiser régulièrement les coordonnées de texture des triangles à leur valeurs d'origines, ce qui a pour effet indésirable de produire des sauts dans l'animation. On peut alors utiliser deux textures qui seront fondues (*blend*) selon une certaine pondération pour la création de l'image finale. Leur déformation respective est décalée dans le temps, la texture la moins déformée ayant la plus grande pondération.

5.4 Spot noise

La création d'animations avec la technique Spot Noise s'obtient naturellement en intégrant la position des points (spots) le long du flux d'une image à l'autre [58]. Pour les flux stationnaires, l'algorithme produit des animations parfaitement cycliques, ce qui permet de visualiser une animation continue avec un nombre réduit d'images. La phase d'initialisation de l'algorithme consiste à choisir le nombre N d'images qui constitueront l'animation et à assigner un certain nombre de spots d'intensité et de positions aléatoires à chaque image de l'animation. L'image à laquelle est initialement assigné un spot correspond à l'*image centrale* de ce spot. L'intensité des spots est maximale dans leur image centrale. Pour un spot donné, sa position est intégrée avec un pas de temps positif dans les $N/2$ images suivant son image centrale. Si la dernière image de l'animation est atteinte, sa position est intégrée et injectée dans la première. La position d'un spot dans les images placées avant son image centrale est intégrée avec un pas de temps négatif. Ainsi, la moitié des images comporte des spots dont la position a été obtenue par intégration positive, l'autre moitié par intégration négative. Il est à noter que les deux spots obtenus à la fin des intégrations positive et négative à partir du même spot initial, se retrouvent dans deux images successives avec évidemment des positions différentes. Pour supprimer l'effet de saut que produirait la disparition et l'apparition de ces paires de spots, l'intensité de chaque spot varie au cours de l'animation. Elle est maximale à l'image centrale et décroît linéairement vers zéro pour les images de part et d'autre de l'image de référence. Ainsi la transition entre les spots se trouvant à l'opposé de leur image centrale est discrète car correspondant à leur intensité minimale.

5.5 LIC

Dès l'article original présentant LIC, les auteurs décrivent comment animer une telle texture de façon à faire apparaître l'orientation du champ de vecteurs en plus de la direction. Le mouvement est introduit en utilisant comme noyau de convolution un filtre périodique suivant une méthode décrite par Freeman [19]. L'animation est alors possible en générant une série d'images dans lesquelles la phase des filtres périodiques a été incrémentée. De petites vagues semblent ainsi se déplacer dans l'image dans le sens du flux.

La fonction choisie par Cabral et Leedom comme noyau de convolution est la composition de deux fonctions de Hanning fenêtrées. La première est une sorte de gaussienne qui s'annule de part et d'autre de son support, la seconde, sinusoidale, a la propriété d'être périodique (c'est elle qui introduit l'effet de vague). La fonction *k* obtenue est un filtre passe-bas périodique qui peut être exprimé comme suit :

$$k(w) = \frac{1 + cos(cw)}{2} \times \frac{1 + cos(dw + \beta)}{2}$$

où *c* et *d* sont des constantes définissant la dilatation des deux fonctions de Hanning et β est la phase, exprimée en radians. La période de la fonction est 2π. La série d'images sera créée en calculant les coefficients de pondération avec des valeurs de β croissantes prises dans l'intervalle $[0;2\pi]$. L'animation obtenue est parfaitement cyclique.

Lors de l'extension de l'algorithme LIC aux grilles curvilignes, Forsell doit tenir compte des compressions et dilatations des cellules de la grille cartésienne lors du retour dans l'espace physique [17][18]. Cette transformation affecte en effet la longueur des supports du noyau de convolution. Ainsi dans les régions de l'espace physique où les cellules ont été compressées, les vagues apparaissent larges et semblent se déplacer lentement. Le contraire est observé dans les cellules dilatées. Pour supprimer ces défauts, la longueur du support du noyau de convolution (*L*) est choisie en fonction de la densité des cellules dans l'espace de calcul. Le mode de calcul de la densité de la grille est détaillé dans l'article original [17]. L'animation des textures LIC plaquées sur des surfaces curvilignes apparaît ainsi régulière en tout point de la surface.

Forsell tente aussi d'étendre LIC de façon à visualiser l'intensité des vecteurs vitesse lors de l'animation, en plus de la direction et de l'orientation du flux. Pour introduire des vitesses variables, Cabral et Leedom suggéraient de contrôler la fréquence du filtre (*w* dans l'équation précédente) en fonction des vitesses locales. Malheureusement, l'intervalle des vitesses visuellement représentable de cette façon est trop étroit. Forsell propose alors de faire varier la taille de l'incrément de la phase β du filtre. Ainsi, pour chaque pixel, l'incrément de phase de sa fonction de convolution pourra varier de 0 à 90°. Un petit (resp. grand) incrément sera choisi pour un pixel se trouvant dans une région de faible (resp. grande) vitesse.

Cependant, les animations avec vitesses variables obtenues par Lisa Forsell ne sont plus cycliques, des sauts apparaissant lors du passage de la dernière à la première image de l'animation. De plus, la façon dont elle propose de construire les textures ne garantit pas la conservation de corrélation entre des pixels voisins portés par une même streamline. Cette perte de corrélation introduit des problèmes d'aliassage spatio-temporel qui dégradent la qualité de l'animation.

Stalling et Hege ont adapté leur algorithme LIC accéléré et sont parvenus à produire, à partir d'une série de 2N images dont les noyaux de convolution ont été corrélés, une animation cyclique de N images [52].

Au cours de cette thèse, nous avons développé une méthode originale permettant d'animer des champs de vecteurs 2D stationnaires. La représentation obtenue est une texture d'apparence proche de celle produite par LIC et contenant sur une seule image toutes les informations nécessaires à son animation. Nous avons appelé *Motion Map* cette structure de données originale [31]. Les animations sont parfaitement cycliques et permettent de visualiser les vitesses variables. Notre méthode est présentée au chapitre 4.

6 Extension aux flux non stationnaires

La visualisation de champs de vecteurs non stationnaires devient un domaine de recherche de plus en plus crucial car les phénomènes étudiés sont souvent, par nature, non stationnaires. La panoplie de techniques disponibles est beaucoup moins étoffée que pour les champs stationnaires et toutes ont pour le moment une capacité d'expression assez restreinte. La difficulté supplémentaire vient de l'évolution temporelle des données. La création d'animations devient nécessaire, impliquant des problèmes de conservation des cohérences spatiale et temporelle.

Les sections suivantes exposent les techniques spécialement conçues pour les champs non stationnaires ainsi que les extensions des techniques présentées précédemment.

6.1 Systèmes de particules

L'extension de l'animation de particules à des flux non-stationnaires est immédiate. Seule la méthode d'intégration de la position des particules d'un pas de temps à l'autre doit être adaptée.

Ce type de visualisation permet de voir la destination de particules lâchées à partir d'une certaine région du domaine et aussi d'apprécier les effets de concentration et de turbulence du flux. La principale difficulté, inhérente à l'utilisation de particules, réside dans le maintient d'une couverture relativement uniforme sur le domaine à visualiser.

6.2 Intégration de trajectoires dans des champs de vecteurs non-stationnaires

Expérimentalement, les trajectoires visualisées dans de telles conditions sont les *pathlines*, obtenues en photographiant des particules avec un temps d'exposition élevé, les *streaklines*, courbes formées par un colorant injecté de façon continue en un point donné, ou les *timelines*, résultat de la déformation d'une ligne de colorant injectée à un instant donné. Les *streamlines*, dans le cas des flux non-stationnaires, ne peuvent pas être obtenues expérimentalement puisqu'elle représentent des courbes tangentes au champ de vecteurs à un instant donné. Elle sont par contre facilement calculées numériquement et leur visualisation montre l'évolution de la structure instantanée du flux. Dans une étude comparative [38], Lane montre la plus grande aptitude des streaklines et timelines, par comparaison aux streamlines, pour visualiser l'apparition et l'effondrement des turbulences dans les flux non-stationnaires (voir figure 2.15).

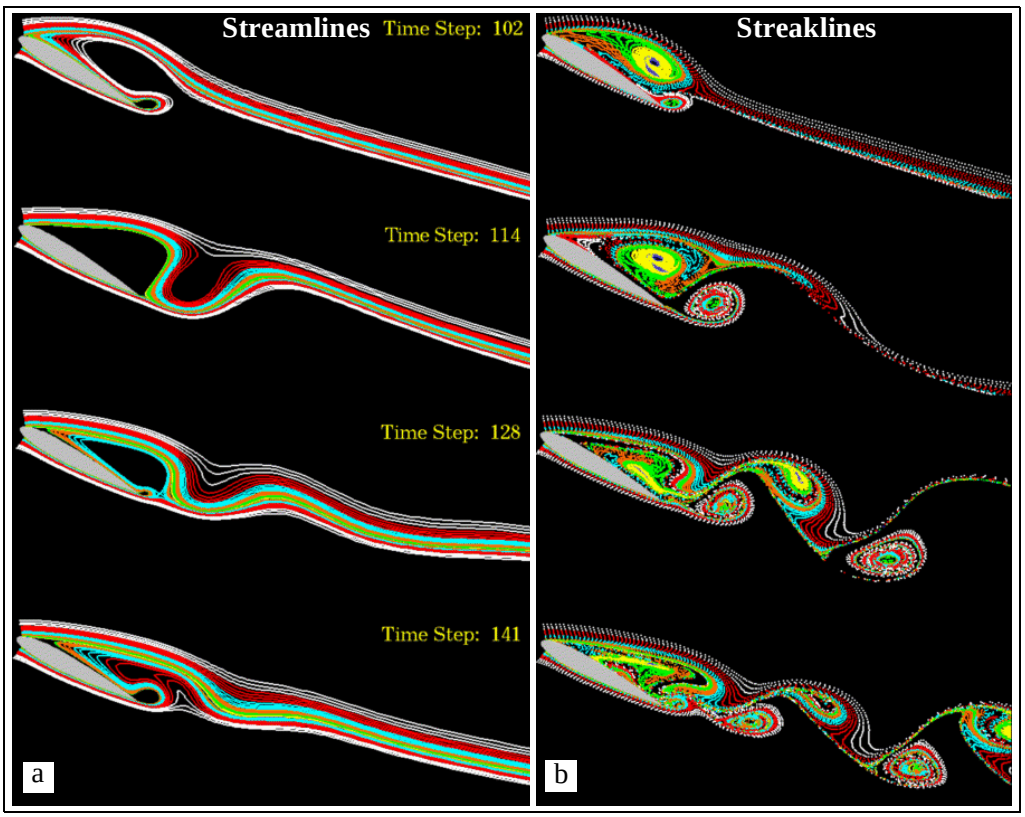

Figure 2.15 Visualisation des turbulences à l'arrière d'une aile d'avion. Les streaklines (b) sont mieux adaptées que les streamlines (a) pour la mise en évidence des turbulences. (images extraites de [38])

Une grande part des travaux portant sur ce domaine consiste en l'amélioration des algorithmes d'intégration numérique permettant d'obtenir ces trajectoires. En plus du fait qu'il faut passer d'un champ de vecteurs à l'autre à chaque pas d'intégration, des solutions spécifiques doivent être apportées quand les champs de vecteurs successifs sont définis sur des grilles différentes, ce qui est courant avec les méthodes de simulation utilisées (Lane [37]). Pour limiter les erreurs d'intégration dues aux discontinuités des grilles, le calcul des trajectoires est souvent fait directement dans l'espace physique. Un mécanisme de recherche efficace des cellules dans lesquelles passent successivement les particules a été proposé (Kenwright [33]).

Peu d'objets construits à partir de trajectoires (voir section 3.3) ont été étendus au cas des flux non-stationnaires. Les *flow-volumes* font partie de ces rares extensions (Becker [4]). Dans ce cas l'analogie des streaklines a été choisie : la totalité du maillage généré à un instant donné est soumis à la déformation pour le passage au pas de temps suivant. Un mécanisme de subdivision adaptative du maillage doit être mis en place afin d'éviter qu'il ne dégénère en des polygones trop étirés ou compressés. Lors de la visualisation, c'est toute la structure constituée par les polygones qui se déforme sous l'action du flux.

Un inconvénient avec ce type d'approche par trajectoires est la difficulté à les initialiser pour que leur emplacement au bout de quelques pas de temps soit encore situé dans une région d'intérêt. Il se pose encore le problème du placement des sources d'injection de manière à maintenir une bonne couverture du domaine au cours de la simulation.

6.3 Déformation de textures

Max a étendu sa technique de déformation de textures (vue section 5.3) au flux non-stationnaires [45]. L'approche est similaire, à la différence prés que les coordonnées de texture aux sommets du maillage sont maintenant intégrées dans les champs de vecteurs successifs. Le recours au fondu de deux textures est encore nécessaire pour pallier le problème de distorsion excessive qui survient pour chaque texture au bout de quelques pas d'intégration.

6.4 LIC

Une idée simpliste pour créer l'animation d'un flux non stationnaire serait de calculer une texture LIC pour chaque pas de temps en convoluant le même bruit blanc d'entrée. Le résultat n'est malheureusement pas exploitable car les pixels se trouvant à une même position sur toutes les images de l'animation sont très peu corrélés. En effet, les streamlines qui ont servi à leur convolution ont des formes différentes d'un pas de temps à l'autre (et couvrent donc des ensembles de pixels différents).

Les deux tentatives les plus remarquées d'extension de LIC aux champs de vecteurs non-stationnaires sont celles de Forsell et surtout de Shen. Forsell garde le canevas de l'algorithme LIC initial mais remplace les streamlines, supports de la convolution, par des pathlines, pour tenir compte de l'évolution du champ de vecteurs au cours du temps [18]. La valeur d'un pixel à un instant donné est maintenant la convolution des pixels traversés par une pathline initiée en ce premier pixel et intégrée dans les deux sens (comme précédemment pour la streamline). Malheureusement les résultats ne sont guère convaincants et il a été souligné que cette approche n'est pas correcte [49] car la valeur d'un pixel dépend aussi de l'état du flux dans le futur (intégration de la pathline dans les deux sens), ce qui ne correspond pas à la réalité physique.

La meilleure méthode de type LIC est à ce jour UFLIC (*Unsteady Flow Line Integral Convolution*), proposée par Shen [49]. Son algorithme repose sur deux phases qui se chargent respectivement de la cohérence temporelle et de la cohérence spatiale. La cohérence temporelle est assurée par la modification du mécanisme de convolution. Dans l'algorithme LIC original, le résultat de la convolution pour un pixel était obtenu en *rassemblant* et moyennant les valeurs des pixels traversés par une streamline. Dans UFLIC, les supports de la convolution ne sont plus des streamlines mais des pathlines naissant de chaque pixel à chaque pas de temps. Ces pathlines sont intégrées, toujours dans le sens du flux et avec des pas de temps positifs, pendant une certaine durée, qui détermine la longueur du support de la convolution, et la valeur du pixel d'origine est *déposée* dans une liste contenue dans chaque pixel traversé. Le dépôt est daté et le résultat de la convolution pour chaque pixel consiste à moyenner les valeurs de sa liste dont les dates sont inférieures à l'instant courant. La convolution utilisant ce mécanisme de propagation et de dépôt des valeurs des pixels assure la cohérence temporelle des images obtenues. Pour la cohérence spatiale, l'image LIC du pas de temps précédent est utilisée comme texture d'entrée pour le pas de temps courant. Les motifs de l'image précédente se retrouvent déformés dans l'image suivante, mais il est possible de suivre leur transformation progressive. Cette transformation itérative des textures tend à supprimer tout contraste au bout de quelques pas de temps (Okada [46]). Shen filtre donc chaque image de façon à conserver un bon contraste. L'animation ainsi obtenue montre des bandes larges et contrastées, orientées dans la direction du flux dans les régions relativement stables, et des lignes fines, voire floues, dans les régions turbulentes.

Dans le cadre de cette thèse nous avons étendu notre algorithme de placement de streamlines aux champs de vecteurs non stationnaires de façon à voir l'évolution de la structure du flux au cours du temps. En jouant sur la densité de streamlines dans le domaine, nous proposons des représentations tant éparses que denses. Ces travaux sont présentés dans le chapitre 5.

7 Streamlines

Ayant basé nos travaux sur l'utilisation de streamlines, nous terminons ce chapitre en détaillant le mode de calcul de ces courbes particulières. Après avoir défini formellement cet objet, nous montrons comment une streamline peut être construite à l'aide de *pas d'intégration* successifs et nous donnons l'algorithme de l'intégrateur que nous avons utilisé pour les générer.

7.1 Intégration numérique d'une streamline

Pour un champ stationnaire, une streamline est similaire à la trajectoire d'une particule supposée sans masse portée par le flux. Etant donné une fonction $v(p)$ qui donne la valeur du vecteur en tout point p du domaine (par interpolation si nécessaire), cette trajectoire est obtenue en résolvant l'équation différentielle suivante :

$$\frac{dp}{dt} = v(p(t)),\ p(0) = p_0 \tag{2.2}$$

où la fonction $p(t)$ représente la position du point p à l'instant t et p_0 est le point de départ de la trajectoire. La position de la particule aprés un intervalle de temps Δt est déterminée par l'intégrale :

$$p(t + \Delta t) = p(t) + \int_t^{t+\Delta t} v(p(t))dt \tag{2.3}$$

De nombreuses méthodes d'intégration numérique ont été proposées pour résoudre l'équation (2.3) [28]. En résolvant cette intégrale pour plusieurs intervalle de temps consécutifs on obtient une suite de points p_k, $0 < k < n$, qui représente la trajectoire discrétisée de la particule.

L'intégrateur dit d'Euler permet de calculer cette suite de points par la formule :

$$p_{k+1} = p_k + hv(p_k) \tag{2.4}$$

où h influence la taille du pas d'intégration et correspond au Δt de l'équation (2.3). Le processus d'intégration de l'équation (2.4) est illustré par la figure 2.16a. Il est possible d'intégrer des points dans le sens opposé des vecteurs contenus dans le champ en spécifiant un pas d'intégration h négatif. Une streamline sera alors la réunion de deux courbes intégrées dans les deux sens (h positif et négatif) à partir du point de départ.

Pour améliorer la précision de l'intégrateur d'Euler, il est possible de passer par un point intermédiaire p'_k (voir aussi figure 2.16b) :

$$\begin{cases} p'_k = p_k + \frac{1}{2} h v(p_k) \\ p_{k+1} = p_k + h v(p'_k) \end{cases} \tag{2.5}$$

Cette méthode d'intégration est un intégrateur de Runge Kutta d'ordre 2, il est aussi courament appelé intégrateur «midpoint», en référence au point intermédiaire.

Nous avons utilisé cet intégrateur pour construire nos streamlines. Il existe cependant des intégrateurs d'ordre plus élevés qui offrent une précision supérieure en prenant plus de points intermédiaires tel que Runge Kutta d'ordre 4 ou supérieur [7].

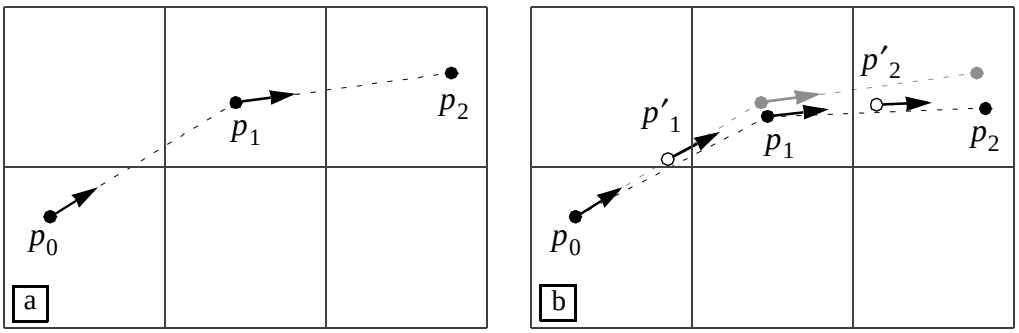

Figure 2.16 Calcul de deux pas d'intégration avec la méthode (a) d'Euler, (b) du Midpoint. Les cercles pleins représentent les points de la streamline, les cercles vides sont les positions intermédiaires utilisées par l'intégration Midpoint. Les flèches partant des cercles sont les valeurs des vecteurs en ces points. L'intégration d'Euler a été superposée en grisé sur la figure b pour montrer la divergence des deux méthodes d'intégration pour un même champ de vecteurs.

Pour le coloriage des streamlines calculées (section 4.1.3 du chapitre 3), notre algorithme nécessitera d'avoir un espacement relativement constant entre les points de chaque streamline, i.e. $\|hv(p'_k)\| \approx \|hv(p'_{k+1})\|$, $0 < k < n-1$. Une approximation suffisante pour notre algorithme a été de normaliser dans une phase d'initialisation chaque vecteur du champ à visualiser. Ainsi en toute position p pour laquelle on dispose d'une valeur, $\|v(p)\| = 1$ et par conséquent $\|hv(p)\| = h$. Toutefois pour les valeurs interpolées $\|v(p)\| \leq 1$ et $\|hv(p)\| \leq h$. Cependant, étant donné la continuité des champs de vecteurs, la norme des vecteurs interpolés reste proche de 1, sauf à proximité d'un point critique où la norme tend vers 0. Nous considérons que la streamline a atteind un point critique, qui signifie l'arrêt de l'intégration, si $\|v(p)\| \leq v_{seuil}$, v_{seuil} étant fixé en pratique à 0.7. Dans ces conditions, $0.7 < \|hv(p'_k)\| \leq 1$, $0 < k < n$, ce qui représente une distance entre les points de la streamline suffisamment constante pour notre algorithme.

Une précision plus importante pourrait être atteinte en utilisant des intégrateurs d'ordre plus élevé fournissant des *sorties denses* (à intervalles réguliers quelque soit la taille du pas d'intégration) comme l'intégrateur DOPRI5 [28] ou un intégrateur de Runge Kutta d'ordre 4 muni d'une interpolation cubique de Hermite [52].

7.2 Algorithme de l'intégrateur

L'algorithme de l'intégrateur que nous avons utilisé pour les champs de vecteurs 2D est décrit par la fonction **Intègre**, écrite en pseudo-code :

```
Fonction Intègre(V, p1, sens)
     entrée :  Champ de vecteurs V
               point p1
               Constante sens égale à AVANT ou ARRIERE
               (comme paramètres globaux : h, v2seuil)
     sortie :  point p2 ou pNull si un point critque est atteind

Si sens = AVANT Alors
| hh = h
Sinon
| hh = -h
FinSi

pp1.x := p1.x + 0.5*hh*Get_VX(p1)
pp1.y := p1.y + 0.5*hh*Get_VY(p1)

Si !DansDomaine(pp1) Alors Renvoyer pNull FinSi

p2.x := p1.x + hh*Get_VX(pp1)
p2.y := p1.y + hh*Get_VY(pp1)

dx := (p2.x - p1.x)
dy := (p2.y - p1.y)
norme2 := dx*dx + dy*dy

Si norme2 > v2seuil et DansDomaine(p2) Alors    // v2seuil est le carré de vseuil
| Renvoyer p2
Sinon
| Renvoyer pNull
FinSi
```

Les fonctions **Get_VX(***p***)** et **Get_VY(***p***)** donnent, par interpolation si nécessaire, les composantes du vecteur à la position *p*. La fonction **DansDomaine(***p***)** renvoie *vrai* si le point *p* se trouve à l'intérieur du domaine.

L'intégration d'une streamline complète est effectuée en appelant plusieurs fois cette fonction sur des positions successives. Le nombre d'appels dépend de la stratégie de construction des streamlines. Nous donnons l'algorithme permettant d'intégrer une streamline complète à la page page 3-42 après avoir spécifié nos contraintes pour cette opération.

8 Conclusion

Nous venons de présenter dans ce chapitre les principales techniques de visualisation de champs de vecteurs disponibles à ce jour dans la littérature. Nous pouvons constater la grande variété des modes de représentation proposés ainsi que la diversité des algorithmes utilisés pour visualiser une donnée commune : un champ de vecteurs.

Parmi les représentations obtenues, nous pouvons distinguer les représentations éparses et denses. Les représentions éparses ont tendance à donner une information locale sous forme de primitives graphiques réparties dans le domaine. Une vue d'ensemble du champ de vecteurs impose alors à l'observateur une interpolation mentale des informations disponibles. C'est typiquement le cas des méthodes utilisant des icônes ou des objets sur quelques trajectoires, comme les flèches ou les streamballs par exemple. Un avantage des représentations éparses est la possibilité d'attirer l'attention de l'observateur sur quelques caractéristiques en des positions précises du domaine. Les représentations denses tirent parti de leur grande résolution spatiale pour donner une information en tout point du domaine. Ce sont typiquement les méthodes de synthèse de texture telles que LIC et SpotNoise. Remarquons enfin que toutes les techniques présentées sont spécialisées pour un type de représentation donné et qu'aucune ne permet l'obtention d'un éventail de représentations allant de éparse à dense.

Nous avons choisi de baser notre travail sur l'utilisation des streamlines. Nous montrons dans cette thèse que ces «objets» vont permettre de générer toutes sortes de représentations, tant éparses, sous forme d'un placement uniforme de trajectoires ou d'icônes portées par ces trajectoires, que denses, sous forme de textures spécialement conçues pour créer des animations de haute qualité. Cette même approche nous a aussi permis d'étendre nos travaux aux champs de vecteurs non stationnaires, là aussi tant en représentation éparse que dense.

Les 3 chapitres suivants décrivent les travaux réalisés pendant cette thèse. Nous aborderons :

- la création de représentations statiques, éparses ou denses, par le placement de streamlines uniformément réparties dans le domaine,
- l'animation de représentation dense de champs de vecteurs stationnaires permettant de visualiser de façon naturelle l'orientation du flux en tout point,
- l'extension de nos algorithmes aux champs de vecteurs non stationnaires.

CHAPITRE

3

Placement de streamlines

1 Introduction

Le tracé de streamlines a été l'un des premiers outils utilisés pour visualiser l'information directionnelle d'un champ de vecteurs. Du fait de leur utilisation très fréquente, un grand nombre de travaux ont porté sur l'amélioration de la précision du calcul des streamlines ou son accélération. Cependant, les premiers travaux visant à améliorer la qualité visuelle des représentations obtenues n'ont été proposés qu'en 1996, avec l'algorithme de placement de Turk et Banks [55]. Auparavant, leur tracé était effectué à partir de points provenant d'un échantillonnage spatial du domaine. La figure 3.1a donne un exemple d'une telle représentation, dont les streamlines ont été initiées depuis un ensemble de points disposés sur une grille régulière couvrant le domaine à étudier. On constate que des régions apparaissent surchargées en streamlines alors que d'autres en sont démunies. L'oeil de l'observateur se trouve alors attiré par les régions plus contrastées, sans que ce contraste soit porteur d'information.

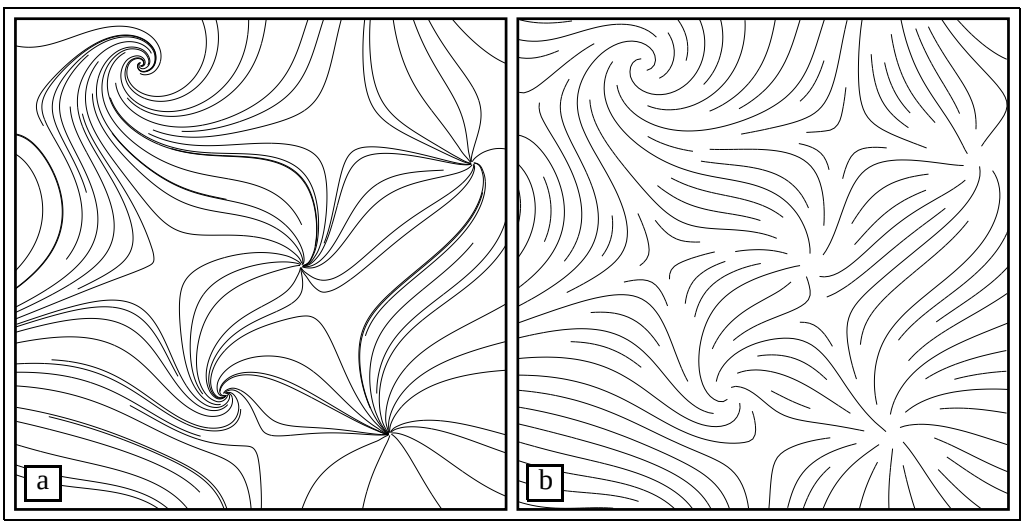

Figure 3.1 **Deux stratégies de placement**. (a) streamlines initiées à partir de points alignés sur une grille régulière. (b) notre algorithme de placement de longues streamlines respectant un critère de densité donné.

Le but d'un algorithme de placement est de tendre vers une représentation dans laquelle la densité de streamlines est uniforme afin de produire des images de bonne qualité visuelle. C'est dans cet objectif que nous avons développé un algorithme de placement permettant de contrôler efficacement la densité de streamlines dans le domaine (voir figure 3.1b). L'algorithme est rapide et permet de générer un éventail de représentations allant de éparse à dense.

En plus de la densité, un facteur influençant la qualité des représentations est la longueur des streamlines. La structure du champ de vecteurs est plus facilement interprétable sur la figure 3.2b alors qu'elle a la même densité que la figure 3.2a. L'allongement des streamlines réduit l'interpolation mentale que doit faire l'observateur afin de relier les petites streamlines le long du flux.

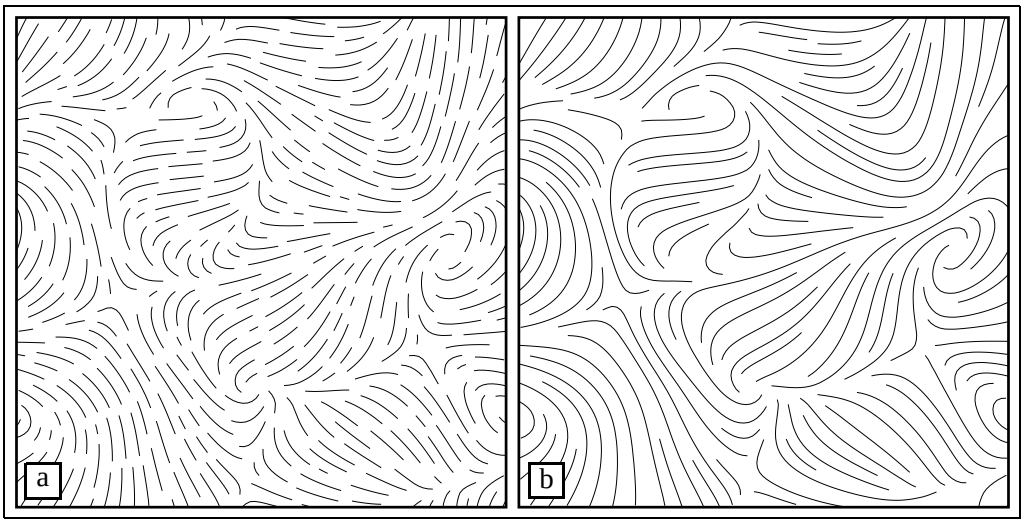

Figure 3.2 **Influence de la longueur des streamlines** sur l'interprétation de la structure du champ de vecteurs. Les deux représentations ont la même densité de streamlines mais elles ont été construites avec des streamlines (a) courtes et (b) les plus longues possibles.

Dans ce chapitre nous allons décrire dans un premier temps l'algorithme permettant d'obtenir de telles représentations (section 2). Nous détaillerons en particulier la façon dont le contrôle de la densité est effectuée et les différentes stratégies de sélection des points de départ des streamlines que nous avons testé. Nous présentons dans la section 3 une extension directe de notre algorithme permettant de produire des ensembles de streamlines multirésolutions. Dans la section 4, plusieurs améliorations du rendu visuel des streamlines sont proposées. Nos résultats sont comparés en section 5 à ceux disponibles dans la littérature et la section 6 nous permet de conclure ce chapitre.

2 Description de l'algorithme de placement

Notre algorithme a été conçu de telle sorte que l'ensemble des streamlines placées dans le domaine respecte un critère de densité donné. Nous avons traduit ce critère global de densité comme une condition locale sur la distance entre les streamlines. La densité souhaitée est alors atteinte si les streamlines ne se rapprochent pas au delà d'une distance minimale et si aucune nouvelle streamline ne peut être insérée en quelque endroit du domaine sans franchir cette distance minimale.

L'algorithme que nous proposons pour obtenir un tel ensemble de streamlines peut être décrit de la façon suivante : à un instant donné, un point de départ d'une streamline est choisi suffisamment éloigné de celles déjà construites, la nouvelle streamline est alors intégrée jusqu'à ce que ses extrémités sortent du domaine, rencontrent un point critique (source ou puits) ou se rapprochent trop près d'une autre streamline. Ce processus est itéré jusqu'à ce qu'aucune nouvelle streamline ne puisse être insérée dans le domaine.

Nous allons aborder dans les sections suivantes trois aspects de la mise en oeuvre de cet algorithme, à savoir :

- le contrôle de la *distance de séparation* entre les streamlines,
- l'intégration des streamlines,
- la méthode de sélection des points de départ des streamlines.

Après avoir proposé les alternatives possibles et justifié nos choix pour ces trois aspects, nous présenterons l'algorithme de placement ainsi que les résultats obtenus.

2.1 Contrôle de la distance de séparation

Ce contrôle tient un rôle prédominant dans notre algorithme puisqu'il garantit de ne pas dépasser la densité souhaitée de streamlines. On exprime cette densité sous forme d'une distance de séparation d_{sep} entre les streamlines. Le contrôle consiste alors à vérifier pendant la construction de toute nouvelle streamline que la distance entre son extrémité courante et les streamlines déjà placées est supérieure à d_{sep}.

Les streamlines sont obtenues par intégration numérique (voir section 7 du chapitre 2) et sont composées d'une série de points appelés *sample points*. A chaque pas de l'intégration, une streamline est prolongée en calculant la position d'un nouveau sample point. Ce sample point P sera alors dit *valide* dans le cadre de notre algorithme si sa distance avec les streamlines existantes est supérieure à d_{sep}.

Les sections suivantes décrivent la façon dont est évaluée la distance entre tout nouveau sample point et les streamlines existantes (sections 2.1.1 et 2.1.2) et

comment le *test de validité* d'un sample point est accéléré afin de rendre notre technique de placement plus efficace (section 2.1.3). L'algorithme du test de validité est alors présenté dans la section 2.1.4.

2.1.1 Calcul de la distance entre un point et une streamline

Une streamline est une série de sample points reliés par des segments de droite. Evaluer la distance entre un sample point candidat *P* et une streamline revient alors à calculer la distance entre un point et une ligne brisée. Cette distance *d* se calcule en prenant la distance minimale entre *P* et le segment le plus proche de la ligne brisée (voir figure 3.3).

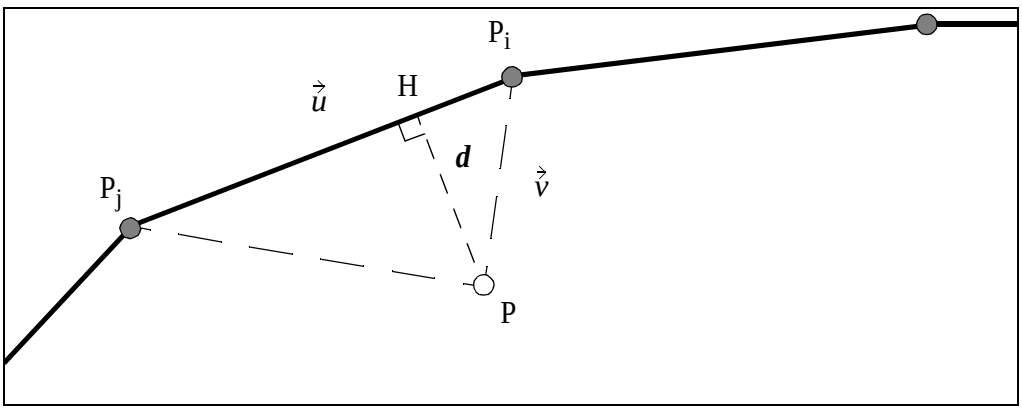

Figure 3.3 Soit d la distance du point P au segment P_iP_j

Considérons un segment P_iP_j dont les coordonnées des sommets sont $P_i = (x_{P_i}, y_{P_i})$ et $P_j = (x_{P_j}, y_{P_j})$. Soit P_i le sommet le plus proche de P et soient $\vec{u}$ et $\vec{v}$ les vecteurs définis par :

$$\vec{u} = \overrightarrow{P_iP_j} = (x_{P_j} - x_{P_i}, y_{P_j} - y_{P_i}) = (x_u, y_u)$$

$$\vec{v} = \overrightarrow{P_iP} = (x_P - x_{P_i}, y_P - y_{P_i}) = (x_v, y_v)$$

alors la distance d entre le point P et le segment P_iP_j est $d = dist(P, P_iP_j)$ avec :

$$dist(P, P_iP_j) = \begin{cases} \dfrac{\sqrt{(x_v y_u^2 - x_u y_u y_v)^2 + (y_v x_u^2 - x_u y_u x_v)^2}}{x_u^2 + y_u^2} & si \;\; x_u x_v + y_u y_v > 0 \\ \sqrt{x_v^2 + y_v^2} & si \;\; x_u x_v + y_u y_v \leq 0 \end{cases}$$

2.1.2 Accélération du calcul de la distance

Il serait coûteux et surtout inutile d'effectuer le calcul de distance entre P et tous les segments des streamlines. En effet, les segments concernés par ce calcul sont ceux dont un des sample points se situe à moins d'une certaine distance d_{seuil} de P. Cette distance dépend de la distance de séparation d_{sep} souhaitée entre les streamlines et de l'écartement $d_{écart}$ entre les sample points consécutifs des streamlines. Nous verrons dans la section 2.2 que nous avons choisi une méthode d'intégration des streamlines fournissant des sample points ayant des écarts constants. d_{seuil} est définie par le cas limite montré sur la figure 3.4.

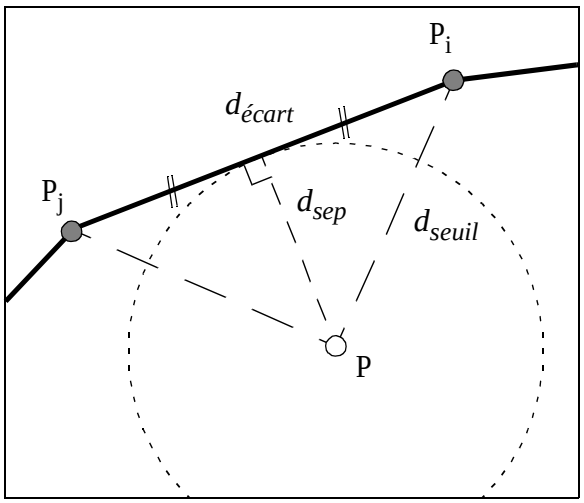

Figure 3.4 Calcul de la distance seuil en fonction de la distance de séparation et de l'écart entre les sample points

La distance seuil est donc :

$$d_{seuil} = \sqrt{d_{sep}^2 + \frac{1}{4} d_{écart}^2}$$

Une première accélération du calcul de distance est donc de ne considérer que les segments dont au moins un des sample points se situe à une distance inférieure à d_{seuil}.

Le calcul peut encore être accéléré dans le cas particulier où la distance de séparation souhaitée d_{sep} est grande devant l'écartement $d_{écart}$ entre les sample points des streamlines. En effet la distance d'un point à un segment est alors proche de la distance du point à son plus proche sommet (voir figure 3.5). Nous faisons alors cette approximation pour accélérer le calcul de distance quand $d_{sep} > 2d_{écart}$. Dans ce cas la distance entre un point et une streamline sera équivalente à la distance minimale entre ce point et les sample points la constituant.

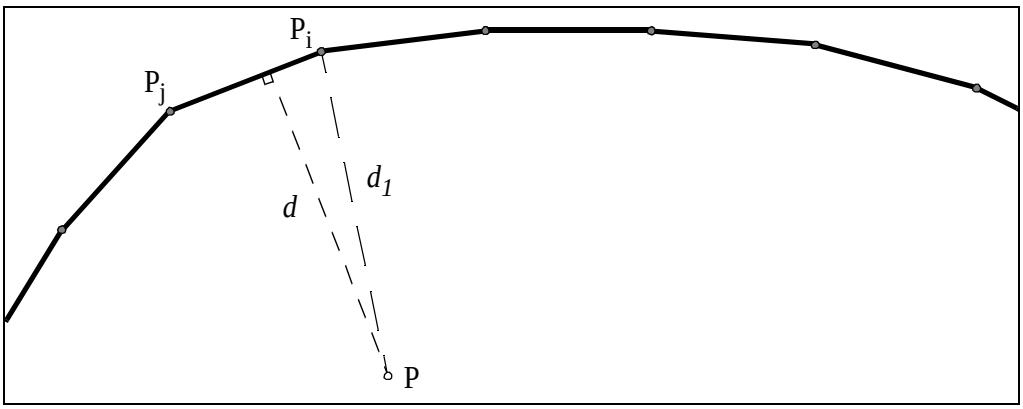

Figure 3.5 Quand P est suffisamment loin de la streamline, la distance d_1 de P à P_i devient une approximation suffisante de la distance d de P au segment P_iP_j.

Les considérations précédentes permettent d'obtenir une bonne approximation de la distance d'un point à un segment d'une streamline tout en rendant le calcul beaucoup plus efficace.

2.1.3 Calcul du test de validité d'un nouveau sample point

L'objectif du contrôle de la densité est de s'assurer que la distance minimale entre un nouveau sample point et toutes les streamlines déjà placées ne descend pas en dessous de la distance de séparation. Pour le vérifier, nous n'avons pas d'autres moyens à ce stade que de parcourir tous les sample points de toutes les streamlines placées et de tester si les distances du point candidat aux segments adjacents à ces sample points sont supérieures à la distance de séparation. La complexité d'un tel algorithme est en $O(N \times M)$ où N est le nombre de streamlines et M le nombre moyen de sample points sur chacune d'entre elles. Comme le test de distance aux streamlines existantes doit être fait pour tout nouveau sample points, il doit être le plus rapide possible. Afin d'accélérer ce test nous avons utilisé une technique de subdivision spatiale, analogue à celle utilisée pour l'accélération du lancer de rayon en synthèse d'image (Arvo [1]). Une grille régulière est superposée au domaine, chaque cellule de la grille référençant tous les points qu'elle contient. La largeur des cellules de la grille est égale à la distance seuil d_{seuil} de sorte que le calcul de la distance d'un point avec toutes streamlines existantes revient à ne considérer que les sample points se trouvant dans les 9 cellules entourant ce point.

A l'utilisation il s'avère qu'en moyenne, seulement 5 à 7 sample points sont considérés lors de la validation d'un sample point candidat et ce indépendamment de la densité. La complexité du test de validité devient ainsi constant et ne dépend plus du nombre de streamlines présentes dans la représentation.

2.1.4 Algorithme du test de validité

Nous avons à présent défini tous les outils nous permettant de contrôler si un nouveau sample point *P* est *valide*, c'est à dire si sa distance par rapport à toutes les streamlines existantes est supérieure à la distance de séparation. Ce test de validité est présenté en pseudo-code dans l'algorithme suivant.

```
Fonction SamplePointEstValide(P, G, d_sep)
     entrée :  Le sample point à tester P
               Grille d'accélération G contenant tous les sample points placés
               La distance de séparation souhaitée d_sep
               (comme paramètres globaux : d_écart et d_seuil)
     sortie :  Vrai ou Faux selon que le sample point est valide ou non

Pour chacune des 9 cellules c de G autour de P
  Pour chaque sample point P_c dans c
    d1 := distance(P, P_c)
    Si d1 < d_sep
      Renvoyer Faux
    Sinon
      Si d1 < d_seuil et d_sep < 2*d_écart
        soit P_p le sample point précédant P_c sur la streamline qui porte P_c
        Si P_p existe
          d2 := dist(P, P_cP_p)
        Sinon d2 := MAX_REEL
        soit P_s le sample point suivant P_c sur la streamline qui porte P_c
        Si P_s existe
          d3 := dist(P, P_cP_s)
        Sinon d3 := MAX_REEL
        d4 := min(d2, d3)
        Si d4 < d_sep
          Renvoyer Faux
        FinSi
      FinSi
    FinSi
  FinPour
FinPour
Renvoyer Vrai
```

2.2 Intégration des streamlines

Dans le chapitre précédent, nous avons détaillé l'opération d'intégration numérique permettant de calculer la position des sample points consécutifs d'une streamline. La longueur de la streamline dépend du nombre de pas d'intégration effectués. Afin de favoriser les longues streamlines nous itérons le processus d'intégration le plus longtemps possible, jusqu'à ce que la streamline sorte du domaine ou rencontre un point critique ou encore que le test de validité pour le nouveau sample point calculé renvoie `faux`. Il est aussi nécessaire de considérer une condition d'arrêt supplémentaire, pour les *streamlines circulaires* : lorsque la trajectoire décrit une boucle dans le domaine. Dans ce cas l'intégration pourrait continuer indéfiniment. On stoppe alors l'intégration quand la distance entre ses extrémités devient inférieure à un certain seuil, que nous avons fixé à $d_{sep}/2$.

Lorsque l'intégration s'arrête pour une des raisons citées ci-dessus, la streamline n'est validée que si sa longueur est supérieure à une longueur minimum. La suppression des petites streamlines à pour effet de favoriser l'apparition de streamlines plus longues en leur laissant plus de place pour croître. Notons enfin que l'inclusion du test de validité du sample point courant dans l'arrêt de l'intégration des streamlines garantit par construction que les représentations ne dépassent pas le critère de densité souhaité, puisqu'aucune streamline ne se rapprochera trop près des streamlines existantes.

L'algorithme suivant décrit le processus d'intégration d'une streamline. Nous utilisons la fonction **Intègre** définit à la page 32. Rappelons que la fonction **Intègre** renvoie le sample point suivant dans la direction qui lui est indiquée (AVANT ou ARRIERE). Si ce sample point est en dehors du domaine ou qu'un point critique a été atteint, la fonction **Intègre** renvoie par convention pNULL.

```
Fonction IntègreStreamline(V, G, p, d_sep)
     entrée :  Champ de vecteurs V
               Grille d'accélération G contenant tous les sample points placés
               Seed point p
               Distance de séparation d_sep
               (comme paramètres globaux : min_longueur)
     sortie :  La streamline intégrée S (valide ou pas)

continueIntégrationAvant   := Vrai
continueIntégrationArrière := Vrai
streamlineCirculaire       := Faux
p est le premier sample point de la streamline S
p_1 := p
p_2 := p
Répèter
  Si continueIntégrationAvant = Vrai
    p_1 := Intègre(V, p_1, AVANT)
    Si p_1 != pNULL et SamplePointEstValide(p_1, G, d_sep)
      Ajouter p_1 à la queue de S
    Sinon continueIntégrationAvant := Faux
  FinSi
  Si continueIntégrationArrière = Vrai
    p_2 := Intègre(V, p_2, ARRIERE)
    Si p_2 != pNULL et SamplePointEstValide(p_2, G, d_sep)
      Ajouter p_2 à la tête de S
    Sinon continueIntégrationArrière := Faux
  FinSi
  // Traitement des streamlines circulaires
  Si Longueur(S) > d_sep/d_écart et distance(p1,p2) < d_sep/2
    streamlineCirculaire := Vrai
  FinSi
Tant que (continueIntégrationAvant =Vrai ou continueIntégrationArrière =Vrai)
         et streamlineCirculaire = Faux

Si Longueur(S) > min_longueur
  la streamline S est valide
  Référencer tous les sample points de S dans la grille d'accélération G
Sinon
  la streamline S n'est pas valide
FinSi
Renvoyer S
```

2.3 Sélection des points de départ des streamlines

Chaque streamline est construite à partir d'un point de départ (appelé *seed point* par la suite, *seed* = graine). L'objectif principal de la méthode de sélection des seed points est de fournir suffisamment de points de départ pour la construction des streamlines de façon à couvrir complètement le domaine.

De la même façon que nous parlions de sample point valide lors de la phase d'intégration des streamlines, nous appellerons *seed point valide* un seed point placé au delà de la distance de séparation de toutes les streamlines existantes. Seuls les seed points valides sont susceptibles de donner naissance à une streamline.

La notion de *couverture complète* est importante car on souhaite pouvoir visualiser la structure de n'importe quel champ de vecteurs sur tout le domaine, sans en avoir de connaissance à priori. Comme le placement des streamlines doit respecter le critère de densité, la couverture complète signifie qu'à la fin du placement, il n'est plus possible d'insérer la moindre streamline en respectant ce critère. La méthode d'intégration des streamlines nous assure déjà qu'on ne dépassera pas la densité de streamlines souhaitée. Un objectif de la technique de sélection des seed points est de s'approcher au plus prés de la densité autorisée, et ce le plus efficacement possible.

Nous avons testé et/ou conçu plusieurs méthodes de sélection et nous avons étudié les paramètres permettant de produire les distributions de streamlines les plus esthétiques possibles.

Dans les sections suivantes nous présentons quatre techniques de sélection de seed points. Les deux premières sont triviales, il s'agit des sélections *sur une grille régulière* et *aléatoire*. Les deux suivantes sont de notre conception : sélection *aléatoire dans un masque booléen* et *dans le voisinage des streamlines déjà placées*. Après avoir décrit le principe de ces techniques de sélection, nous comparons leur efficacité respective dans la section 2.3.5.

2.3.1 Sélection sur une grille régulière

Cette méthode de sélection de seed points a l'intérêt d'être simple à mettre en oeuvre puisqu'il suffit de superposer au domaine une grille dont les noeuds sont proposés comme emplacements des seed points. L'espacement entre chaque noeud doit être adapté à la distance de séparation de façon à favoriser l'insertion du maximum de streamlines tout en minimisant le nombre de noeuds. Les coordonnées des samples points sont générées grâce à deux boucles imbriquées, ce qui rend la méthode particulièrement efficace.

Un avantage important de cette approche est que l'on connaît à priori le nombre de seed points que l'algorithme proposera, ce nombre étant égal à $N_x \times N_y$ points

où N_x et N_y sont respectivement le nombre de colonnes et de lignes de la grille. Dans la pratique, ce nombre est un majorant bien éloigné du nombre suffisant de seed points, car au moment où ils sont testés, certains sample points ont été préalablement recouverts par une streamline. Il faut cependant balayer toute la grille pour être sûr qu'aucune région n'a été oubliée. La figure 3.6 montre les représentations obtenues pour des champs de vecteurs plus ou moins turbulents. Les positions des seed points qui ont donné naissance aux streamlines ont été marqués.

Une faiblesse de cette technique est l'apparition sur la représentation finale de motifs formés par la position des extrémités des streamlines. Ces motifs n'apparaissent toutefois que pour des champs de vecteurs peu turbulents, comme sur la figure 3.6b. La cause de leur apparition est l'ordre régulier du balayage de la grille.

Des parcours moins réguliers de grilles ont été proposés dans la littérature. Il s'agit par exemple des parcours par blocs ou à l'aide de séquences pseudo-aléatoires de type Sobol (Hege [24]). Le but recherché était une couverture maximale du domaine avec un nombre donné de streamlines de longueur fixe pour la génération de textures LIC. Nous avons développé un parcours de grille mieux adapté au placement de streamlines de longueur quelconque, que nous présentons dans la section 2.3.3.

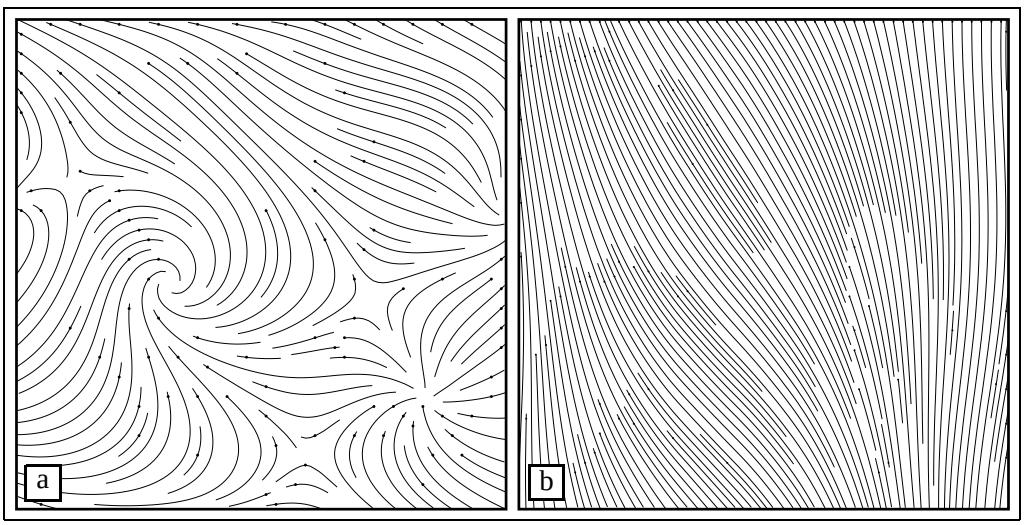

Figure 3.6 Sélection des seed points sur une grille régulière. (a) d_{sep} = 2%, (b) d_{sep} = 1%

2.3.2 Sélection aléatoire

La méthode aléatoire consiste à choisir les seed points au hasard dans le domaine. Un seed point est alors choisi tel que *p = (rand(largeur), rand(hauteur))*, où *rand(x)* est l'appel au générateur de nombres aléatoires retournant un réel entre 0 et *x*.

Ici encore la génération d'un seed point est très peu coûteuse et cette technique présente au moins un avantage sur la précédente : comme des seed points consécutifs ont peu de chance d'être choisis dans une même partie du domaine, il n'y a pas d'apparition de motifs dûs à un balayage régulier, tel que le montre la figure 3.7b.

Cette approche comporte toutefois un certain nombre d'inconvénients. On n'a, par exemple, aucune information à priori sur la *qualité* des seed points fournis. En effet, étant choisi aléatoirement, un seed point peut se trouver trop près d'une streamline. Cette tendance à fournir des seed points non valides va croître avec le remplissage du domaine. Il se pose alors le problème de la terminaison de l'algorithme : quand l'algorithme de sélection aléatoire doit-il arrêter de fournir des seed points ? Une solution consiste à en fournir un nombre supposé suffisant. Une estimation de ce nombre peut être donnée par le nombre de points que fournirait une grille régulière. Un avantage de cette méthode est qu'elle fournit une solution au problème de terminaison sans la mise en place de mécanisme supplémentaire. Bien sûr, sa faiblesse et de n'avoir aucun retour sur la façon dont est couvert le domaine : la génération peut continuer alors que la totalité du domaine est couvert ou au contraire s'arrêter alors que certaines zones ne sont pas couvertes. Contrairement aux autres méthodes présentées ici, cette méthode ne garantit pas une densité de streamlines homogène sur toute l'image mais elle assure seulement qu'aucune zone ne présente une densité supérieure à la densité maximum autorisée (garantit par construction par notre algorithme de contrôle de la distance de séparation).

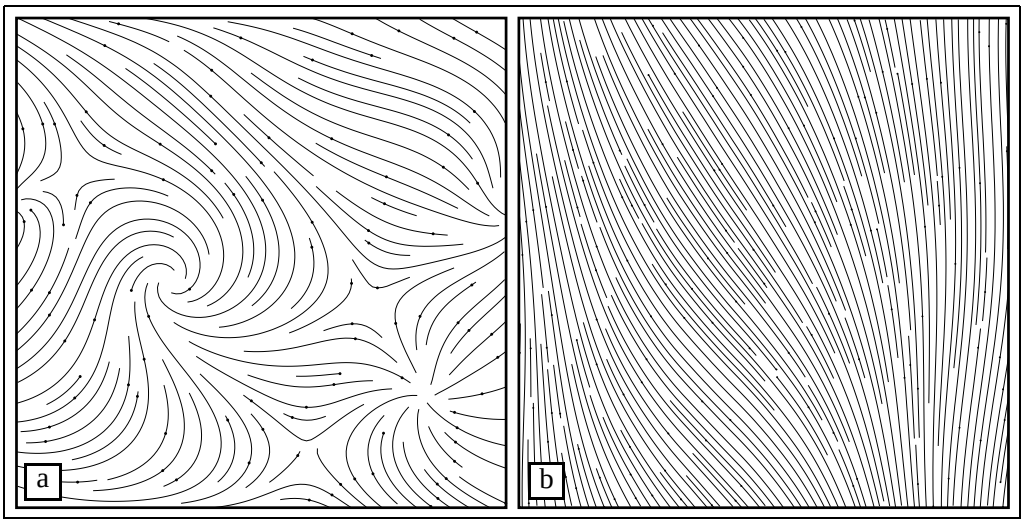

Figure 3.7 Sélection aléatoire des seed points. (a) d_{sep} = 2%, (b) d_{sep} = 1%

2.3.3 Sélection aléatoire dans un masque booléen

Les méthodes proposées auparavant sont très simples à mettre en oeuvre et le coût de génération des seed points est minimal. Elles comportent cependant quelques inconvénients (motifs, problème d'arrêt) qui sont principalement dûs à leur progression «en aveugle», c'est à dire sans tenir compte de l'occupation du domaine par les streamlines déjà placées.

Nous avons développé une technique de sélection de seed points qui présente à elle seule, les avantages des deux précédentes. Cette méthode supprime l'effet de motifs, assure la couverture complète du domaine et garantit la terminaison de l'algorithme. Les seed points sont choisis aléatoirement sur une grille régulière qui sert de surcroît de masque booléen. Une cellule du masque est mise à `faux` dès qu'elle sert à générer un seed point, ce qui évite de fournir plusieurs fois un seed point issu de la même cellule.

Afin de trouver la position d'un seed point, une cellule du masque est choisie aléatoirement. Si sa valeur est `faux`, un parcours du masque de type ligne est effectué jusqu'à ce qu'une cellule *libre* (dont la valeur est `vrai`) soit trouvée. Le centre de cette cellule est proposé comme seed point et sa valeur est mise à `faux`. Si on arrive au coin inférieur droit de la grille, on poursuit par le coin supérieur gauche. L'algorithme termine lorsque qu'un parcours complet de la grille a été effectué sans trouver de cellule libre. A chaque fois qu'un nouveau sample point est calculé, la cellule du masque contenant le point est mise à `faux`. Ce mécanisme permet de prendre en compte l'occupation du domaine en fournissant des seed points se trouvant dans des régions libres. Les représentations obtenues sont visuellement similaires à celles générées par les méthodes précédentes à l'exception des motifs qui n'apparaissent plus (voir figure 3.8). Cependant la gestion du masque et la recherche de cellules libres pénalisent l'efficacité de cette méthode de sélection (voir section 2.3.5).

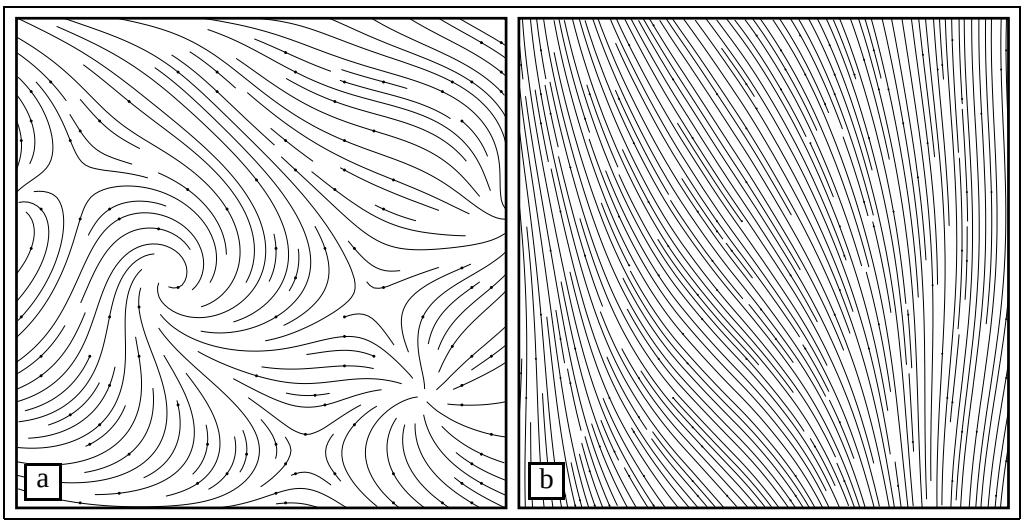

Figure 3.8 Sélection aléatoire dans un masque booléen. (a) d_{sep} = 2%, (b) d_{sep} = 1%

2.3.4 Sélection dans le voisinage des streamlines déjà placées

Nous avons conçu une quatrième méthode de sélection de seed points qui s'est avérée plus efficace que les précédentes. La démarche est radicalement différente puisqu'on s'appuie maintenant sur chaque streamline existante pour créer des seed points dans son voisinage. Le principe est de générer une liste de seed points candidats à partir d'une streamline en calculant les coordonnées de points se trouvant à une distance $d_{seed} > d_{sep}$ de cette streamline.

Il existe plusieurs stratégies pour générer une telle liste. La plus immédiate est de parcourir successivement les sample points de la «streamline génératrice» et de proposer des seed points à une distance d_{seed} de part et d'autre de cette génératrice (voir figure 3.9). Une autre stratégie consiste à parcourir les sample points de façon dichotomique : le seed point du milieu de la streamline est choisi, puis les milieux des deux demi streamlines et ainsi de suite, récursivement. Nous avons observé que pour le cas des champs de vecteurs peu turbulents, le parcours dichotomique produisait des distributions de streamlines globalement plus esthétiques.

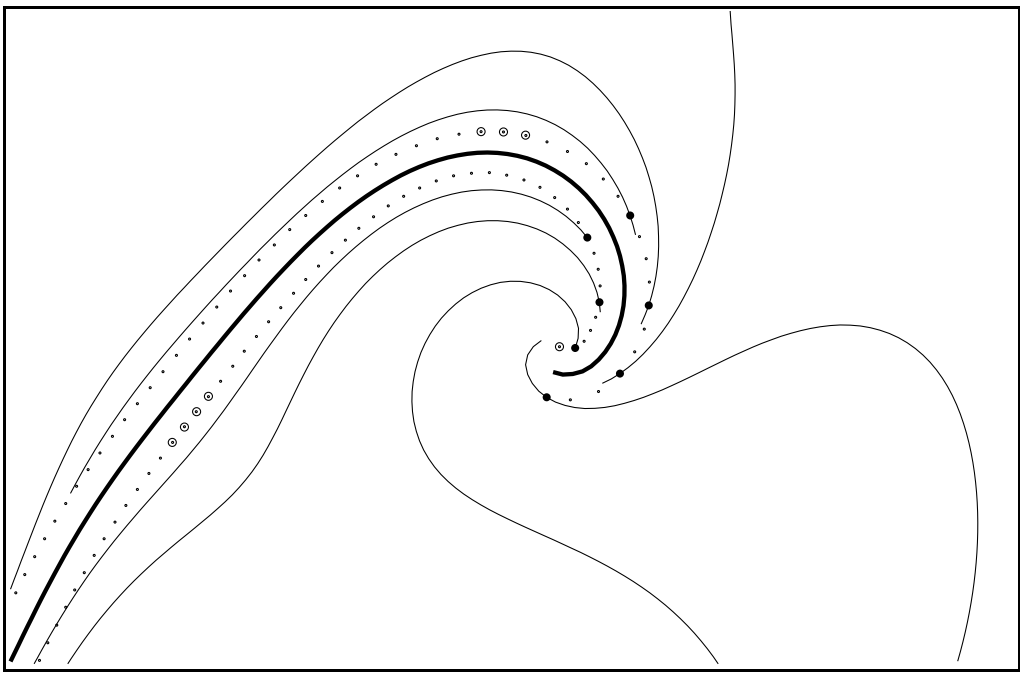

Figure 3.9 Création de seed points candidats en parcourant successivement les sample points d'une streamline. Les seed points candidats entourent la streamline génératrice (ligne épaisse). Les cercles montrent les seed points valides au moment de leur création mais qui n'ont pas fourni de streamlines valides (car trop courtes). Les disque noirs sont les seed points dont sont issues les autres streamlines qui ont été dessinées.

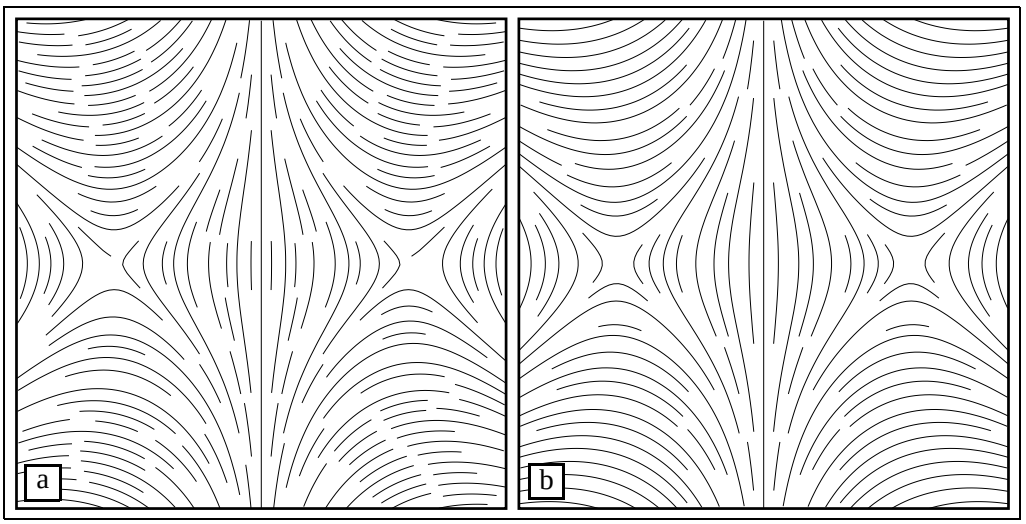

Figure 3.10 Influence du coefficient $\texttt{coef}_{\texttt{sep\_seed}}$ sur la longueur des streamlines générées (a) $\texttt{coef}_{\texttt{sep\_seed}} = 1.05$; (b) $\texttt{coef}_{\texttt{sep\_seed}} = 1.5$

On remarque sur la figure 3.9 que tous les seed points ne donnent pas naissance à une streamline valide mais nous sommes assurés qu'après les avoir tous essayés, aucune autre streamline ne peut être initiée à proximité de la streamline génératrice. Chaque streamline ne sera donc utilisée qu'une fois en tant que streamline génératrice. Nous verrons dans la section 2.4 que cette dernière propriété servira de base à l'élaboration du test de terminaison de l'algorithme de placement.

Nous avons lié d_{seed} à d_{sep} par un coefficient $coef_{sep\_seed}$ qui règle la distance à laquelle sont semées les nouvelles streamlines ($d_{seed} = d_{sep} \times coef_{sep\_seed}$). Ce coefficient doit être supérieur à 1 pour que les seed points se trouvent au delà de la distance de séparation. En choisissant le coefficient suffisamment grand, les seed points sont placés suffisamment loin de la streamline génératrice, ce qui favorise l'obtention de longues streamlines. La figure 3.10 donne un aperçu de l'influence du coefficient $coef_{sep\_seed}$ sur la longueur des streamlines générées. En pratique nous choisissons $coef_{sep\_seed} = 1.3$. La différence entre les distances d_{seed} et d_{sep} est visible sur la figures 3.9 puisque les extrémités des streamlines proches de la streamline génératrice sont à une distance inférieure (d_{sep}) à la distance aux seed points candidats (d_{seed}).

Cette méthode de sélection de seed points se révèle bien adaptée aux champs de vecteurs turbulents ou non (voir figure 3.11) et d'une meilleure efficacité comparée aux autres méthodes de sélection (voir section suivante).

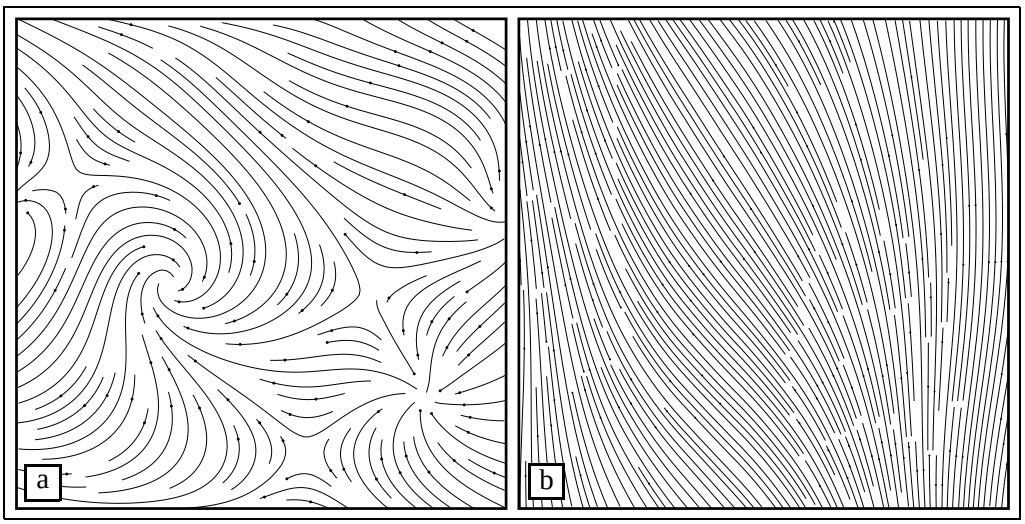

Figure 3.11 Sélection des seed points dans le voisinage des streamlines déjà placées. (a) d_{sep} = 2%, (b) d_{sep} = 1%

2.3.5 Comparaison des techniques de sélection

Nous présentons au début de cette section des éléments permettant de mieux appréhender le fonctionnement des quatre techniques de sélection, nous comparons ensuite leur efficacité respective en terme de temps de calcul.

Un moyen de comprendre l'action des techniques de sélection est d'observer la répartition des seed points qu'elles génèrent. Les figures 3.12a,b et 3.13a,b montrent cette répartition pour les quatre techniques de sélection utilisées pour la création des figures 3.6a, 3.7a, 3.8a et 3.11a. La distance de séparation est la même pour les quatre figures. Sur chaque figure, les disques noirs sont les seed points qui ont donné naissance aux streamlines valides. Les cercles sont les seed points valides qui n'ont pas donné naissance à une streamline valide et les points sont tous les seed points qui ont été proposés par la méthode de sélection.

On reconnaît facilement sur la figure 3.12a la distribution régulière des seed points issus de la grille superposée au domaine, ainsi que leur distribution aléatoire sur la figure 3.12b.

La figure 3.13a montre l'organisation régulière du masque booléen. On remarque que la gestion du masque a permis de fournir beaucoup moins de seed points pour parvenir à une couverture complète du domaine (seulement 35% des cellules du masque ont servi).

Sur la figure 3.13b, la répartition des seed points dans le voisinage des streamlines apparaît nettement.

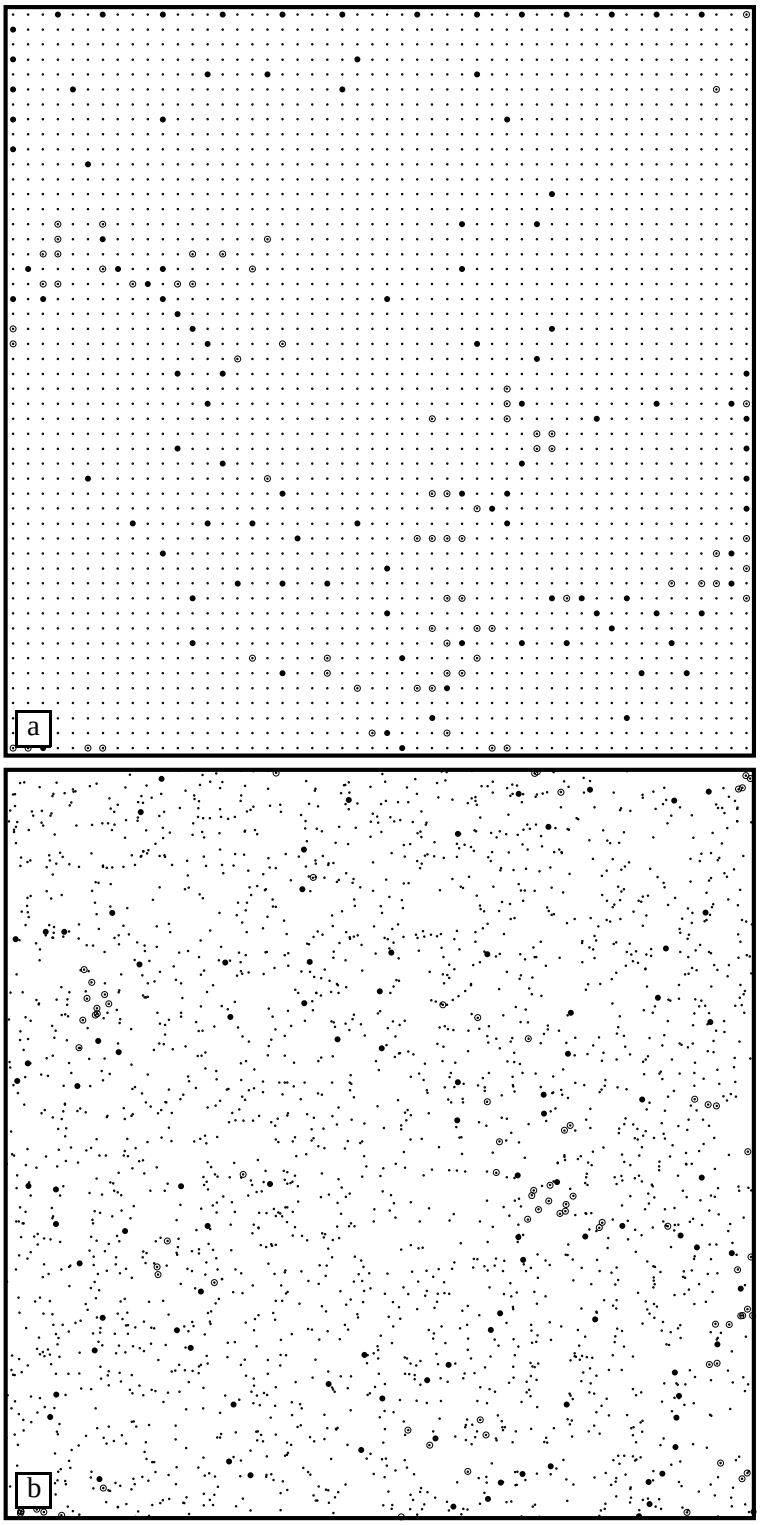

Figure 3.12 Méthodes de sélection de seed points (d_{sep} = 2%). (a) sur grille régulière, (b) aléatoire.

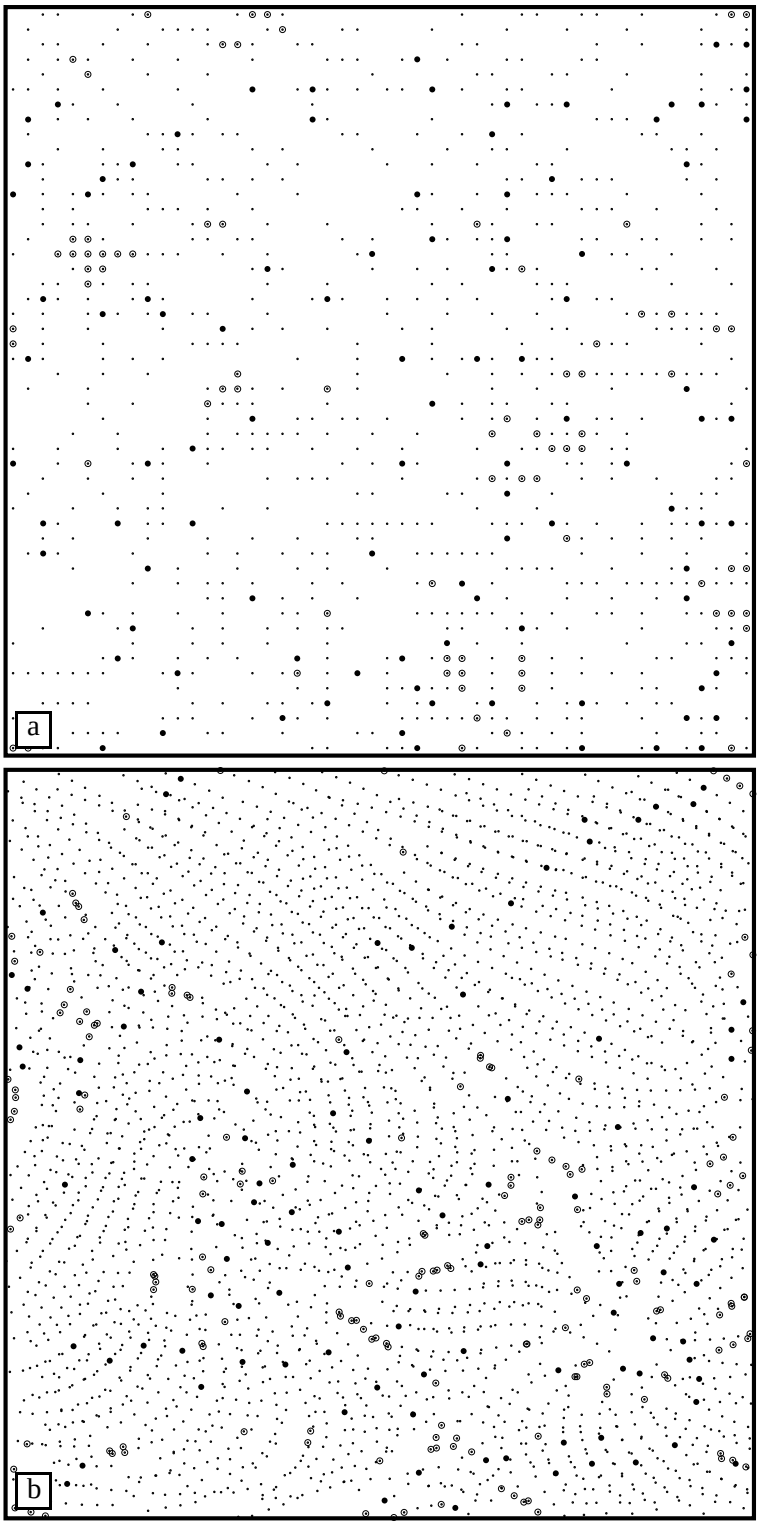

Figure 3.13 Méthodes de sélection de seed points (d_{sep} = 2%). (a) aléatoire sur un masque booléen, (b) dans le voisinage des streamlines déjà placées.

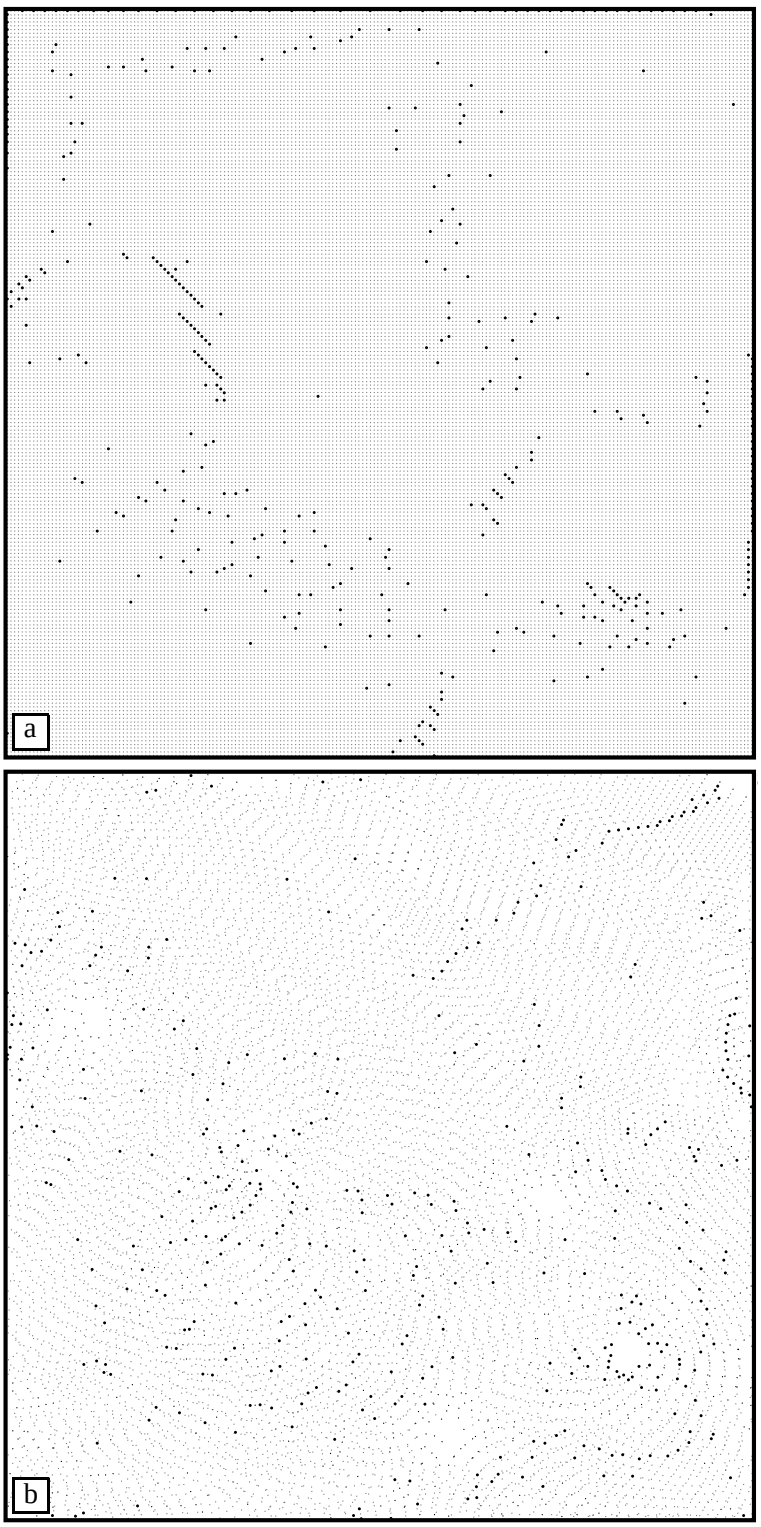

Figure 3.14 Répartition des seed points pour les méthodes de sélection de type (a) *grille* et (b) *voisinage* pour d_{sep} = 0.5%. A cette densité la méthode par voisinage requiert 3 fois moins de seed points que la grille pour couvrir le domaine.

Le tableau 1 et la figure 3.15 donnent les temps d'exécution nécessaires au calcul d'un ensemble de streamlines recouvrant le domaine pour les quatre méthodes de sélection et pour différentes distances de séparation.

Méthode :	**Grille**	**Aléatoire**	**Masque**	**Voisinage**
d_{sep} = 2%	0.47 s *(figure 3.12a)*	0.49 s *(figure 3.12b)*	0.42 s *(figure 3.13a)*	0.61 s *(figure 3.13b)*
d_{sep} = 1%	1.08 s	1.27 s	0.98 s	1.10 s
d_{sep} = 0.5%	2.61 s *(figure 3.14a)*	3.38 s	3.51 s	1.80 s *(figure 3.14b)*
d_{sep} = 0.25%	12.88 s	16.88 s	22.98 s	4.76 s

Tableau 1: Temps de calcul pour les différentes méthodes de sélection de seed points.

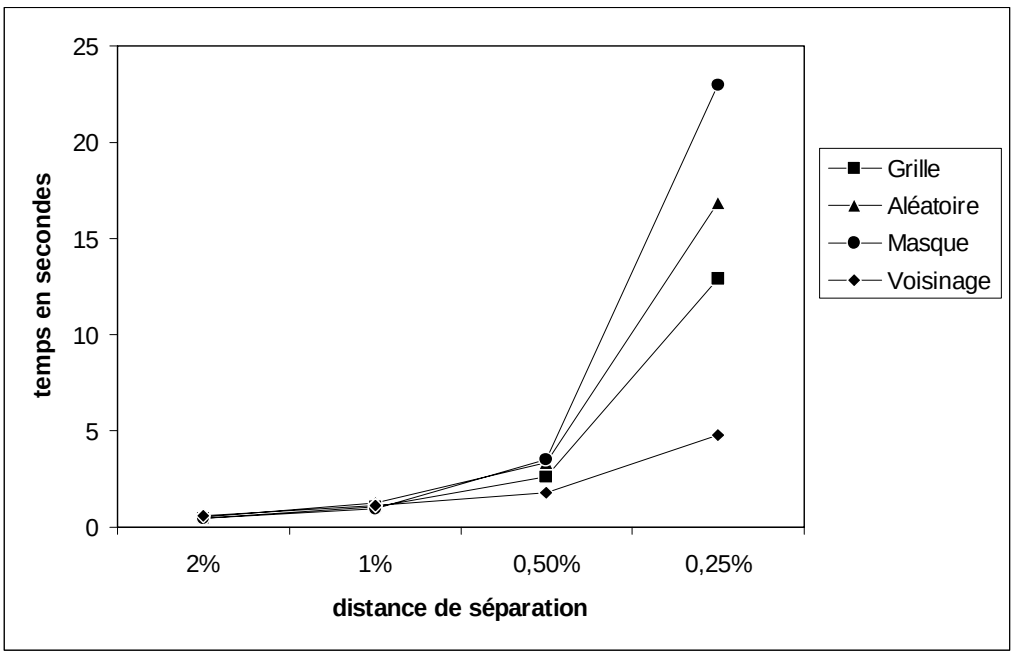

Figure 3.15 Temps de calcul pour les différentes méthode de sélection de seed points.

La méthode de sélection dans le voisinage des streamlines prend nettement l'avantage sur les autres à mesure que la distance de séparation diminue et donc pour des représentations denses. Cette meilleure efficacité s'explique par la plus faible proportion de seed points que fournit la méthode par voisinage par

rapport aux méthodes *aléatoire* et *grille* (voir figure 3.14). La méthode par masque booléen avait aussi été conçue pour diminuer le nombre de seed points proposés, mais le surcoût induit par la gestion du masque pénalise lourdement la méthode.

En plus de son efficacité, un aspect qualitatif intervient en faveur de la méthode de sélection dans le voisinage des streamlines pour les champs de vecteurs peu turbulents. La sélection des streamlines à une distance fixe des streamlines existantes produit de meilleurs résultats visuels que ceux obtenus avec les méthodes aléatoires (voir figure 3.16). En conclusion, cette méthode s'avère nettement meilleure que les précédentes, tant qualitativement que du point de vue du temps de calcul.

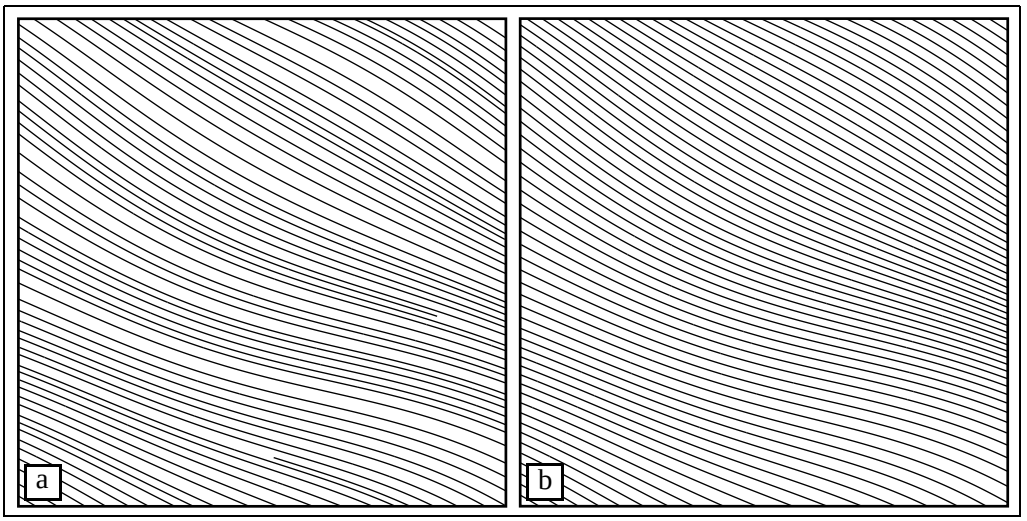

Figure 3.16 Qualité visuelle des images obtenues par placement de streamlines de même densité avec deux méthodes de sélection de seed points : (a) sélection aléatoire, (b) sélection dans voisinage des streamlines.

2.4 Algorithme de placement

Maintenant que sont précisées les méthodes de contrôle de la distance de séparation, d'intégration et de sélection des seed points, nous pouvons assembler ces composants pour former notre algorithme de placement de streamlines. Bien que nous ayons implémenté les quatre techniques de sélection de seed points, nous ne décrirons que l'algorithme de placement avec la sélection des seed points dans le voisinage des streamlines déjà placées. L'inclusion de cette technique de sélection dans l'algorithme de placement est un peu moins triviale que pour les trois autres méthodes et nécessite une description détaillée.

Du fait de la technique de sélection de seed points utilisée, les nouvelles streamlines sont construites à proximité de celles déjà placées, couvrant progressivement tout le domaine. Chaque streamline ne sert qu'une fois au processus de génération de seed points et l'algorithme termine lorsque qu'aucune streamline valide ne peut être construite à partir des seed points générés.

L'algorithme utilise une file F de streamlines dans laquelle sont stockées les streamlines déjà placées mais non encore utilisées en tant que streamline génératrice. A l'initialisation, la file F ne contient généralement qu'une seule streamline valide, mais il est possible de lui en fournir plusieurs et l'algorithme placera les nouvelles streamlines autour de ces streamlines imposées (cette caractéristique sera utile dans le cas du placement de streamlines en multi-résolution - voir section 3). Une streamline extraite de la file sert à générer de nouvelles streamlines qui sont à leur tour placées dans la file. Quand le domaine est pratiquement rempli, très peu de nouvelles streamlines peuvent être placées et la file a tendance à se vider. L'algorithme se termine lorsque que la file est vide (plus de nouvelles streamlines pour l'alimenter). Voici l'algorithme :

```
Fonction PlaceStreamlines(V, L_init, d_sep)
     entrée :  Champ de vecteurs V
               Liste de streamlines d'initialisation L_init
               Distance de séparation d_sep
               (Le coefficient coef_sep_seed comme paramètre global)
     sortie :  La liste de toutes les streamlines placées L_placées

Pour chaque streamline S dans L_init
  | Mettre S dans la file F
  | Mettre S dans la liste des streamlines placées L_placées
FinPour
d_seed := d_sep * coef_sep_seed
fini := Faux
Tant que F n'est pas vide Faire
  | Extraire une streamline S_courante de F
  | Calculer L_sp : liste de seed points à la distance d_seed de S_courante
  | Pour chaque seed point p dans L_sp Faire
  | | S_new := IntègreStreamline(V, L_placées, p, d_sep)
  | | Si S_new est valide Alors
  | | | Ajouter S_new dans F
  | | | Ajouter S_new dans L_placées
  | | FinSi
  | FinPour
FinTantQue
Renvoyer L_placées
```

Cet algorithme a été comparé en terme d'efficacité et de qualité des représentations avec le meilleur algorithme de placement disponible dans la littérature (Turk et Banks [55]). A qualité visuelle similaire les temps d'exécution sont nettement en faveur de notre algorithme. Les résultats sont présentés dans la section 5.

2.5 Distance de séparation variable

La distance de séparation entre les streamlines n'est pas forcement constante et uniforme sur tout le domaine. Il peut être intéressant d'agir sur ce paramètre pour exprimer une caractéristique du flux ou une grandeur extérieure au champ de vecteurs. Ainsi, une convention courante en mécanique des fluides est de représenter des streamlines plus serrées dans les régions où la vitesse du flux est plus grande. L'insertion de cette caractéristique de distance de séparation variable dans notre algorithme est très aisée. Les distances d_{sep}, d_{seed} et d_{seuil} ne sont alors plus de simples paramètres mais des fonctions dont la valeur varie selon la (ou les) grandeur(s) attachée(s) (vitesse du flux, pression...). La figure 3.17 montre un exemple de représentation à distance de séparation variable.

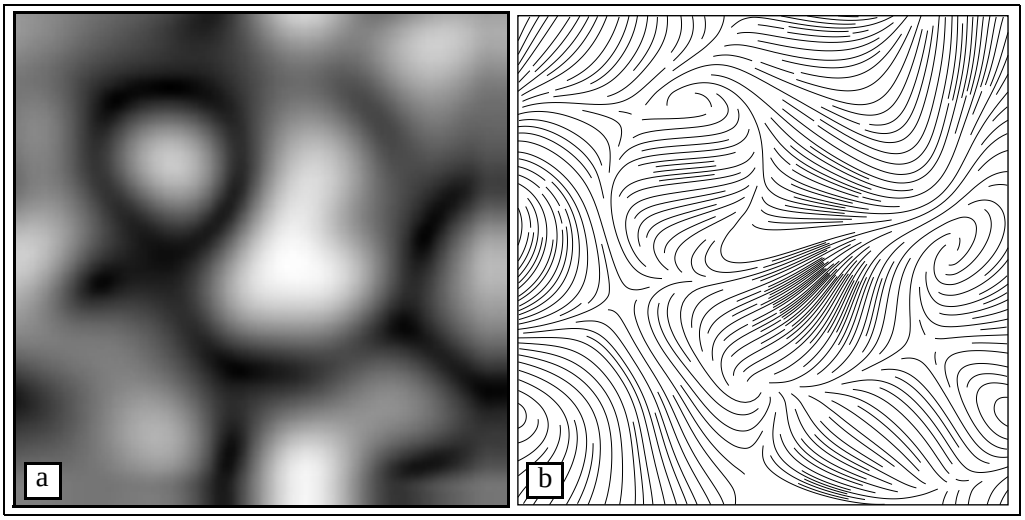

Figure 3.17 **Distance de séparation variable :** d_{sep} est fixée en fonction de la vitesse du flux. (a) carte des vitesses, (b) les streamlines sont plus proches dans les zones de grandes vitesses.

3 Ensemble de streamlines multi-résolutions

La visualisation d'un champ de vecteurs par un ensemble de streamlines peut être étendue en permettant une exploration dynamique du domaine selon deux modes : *enrichissement* et *zoom*. Dans le cas de l'enrichissement, en partant d'une vue d'ensemble ne comportant que quelques streamlines, il est possible d'en ajouter de nouvelles jusqu'à ce que leur nombre précise suffisamment la structure du champ de vecteurs. Il peut être aussi souhaitable de zoomer progressivement vers une région du domaine tout en conservant une densité visuelle de streamlines constante.

Un ensemble de streamlines permettant de telles manipulations doit comporter au moins deux propriétés. D'une part il doit être possible d'afficher les streamlines à une résolution donnée. D'autre part, quelque soit la résolution, les streamlines affichées doivent être placées uniformément dans le domaine (respect du critère de densité à toutes les résolutions).

Un aspect intéressant de notre algorithme de placement est qu'il peut être facilement étendu de façon à produire une hiérarchie d'ensembles de streamlines. Chaque niveau de la hiérarchie correspond à un ensemble de streamlines calculées pour une certaine densité. Nous parlerons d'un ensemble de streamlines multi-résolutions parce que les streamlines d'un niveau i de la hiérarchie appartiennent également à tous les niveaux $j>i$. Le critère de densité est respecté pour chaque niveau de la hiérarchie.

3.1 Extension de l'algorithme de placement

La hiérarchie de streamlines est construite itérativement en complétant un ensemble initial par l'insertion de nouvelles streamlines dont la distance de séparation autorisée diminue à chaque itération. Le premier ensemble de streamlines, correspondant à la résolution la plus basse, est calculé avec une distance de séparation entre streamlines maximum, soit max_d_{sep}. Pour les niveaux suivants, la distance de séparation est diminuée. A chaque niveau, des seed points sont déduits de toutes les streamlines contenues dans les niveaux précédents. Les streamlines générées sont ajoutées à la fin de la liste des streamlines placées $L_{placées}$. L'opération est répétée tant que la distance de séparation courante est supérieure à la distance de séparation minimum min_d_{sep}. La figure 3.18 illustre ce processus.

La diminution de la distance de séparation est obtenue par une loi géométrique ($d_{i+1} = d_i$ * *constante* avec $0 <$ *constante* < 1). Ainsi si k est le nombre de niveaux de résolution souhaité, la constante, appelée *coefficient de réduction* ($coef_{réduction}$) est calculée de la façon suivante :

$$coef_{réduction} = \sqrt[(k+1)]{\frac{min\_d_{sep}}{max\_d_{sep}}}$$

où min_d_{sep} et max_d_{sep} sont les distances de séparation minimum et maximum entre les streamlines.

L'algorithme qui calcule l'ensemble de streamlines multi-résolution est le suivant :

```
Fonction PlaceStreamlinesMultiResolution(V, L_init, min_d_sep, max_d_sep, k)
     entrée :  Champ de vecteurs V,
               Liste de streamlines d'initialisation L_init
               Distances de séparation min_d_sep et max_d_sep
               Le nombre de niveaux k
     sortie :  La liste de toutes les streamlines placées L_placées

coef_reduction = pow(min_d_sep/max_d_sep, 1.0/k)
L_placées := L_init
d_courante := max_d_sep
Tant que d_courante >= min_d_sep
  | L_placées := PlaceStreamlines(V, L_placées, d_courante)
  | d_courante:= d_courante * coef_reduction
FinTantque
Renvoyer L_placées
```

L'exécution de cet algorithme a pour effet de ranger dans la liste $L_{placées}$ les streamlines obtenues avec des distances de séparation décroissantes. Pour afficher un ensemble de streamlines à une résolution donnée, il suffit d'afficher les premières streamlines jusqu'à atteindre la distance de séparation correspondant à la résolution souhaitée (les streamlines sont stockées avec un tableau annexe à k indices contenant le numéro de la dernière streamline de chaque niveau). De plus, à chaque itération, toute nouvelle streamline est construite sous la contrainte d'être placée à une distance minimum de toutes les autres streamlines : le critère de densité est donc respecté à toutes les résolutions.

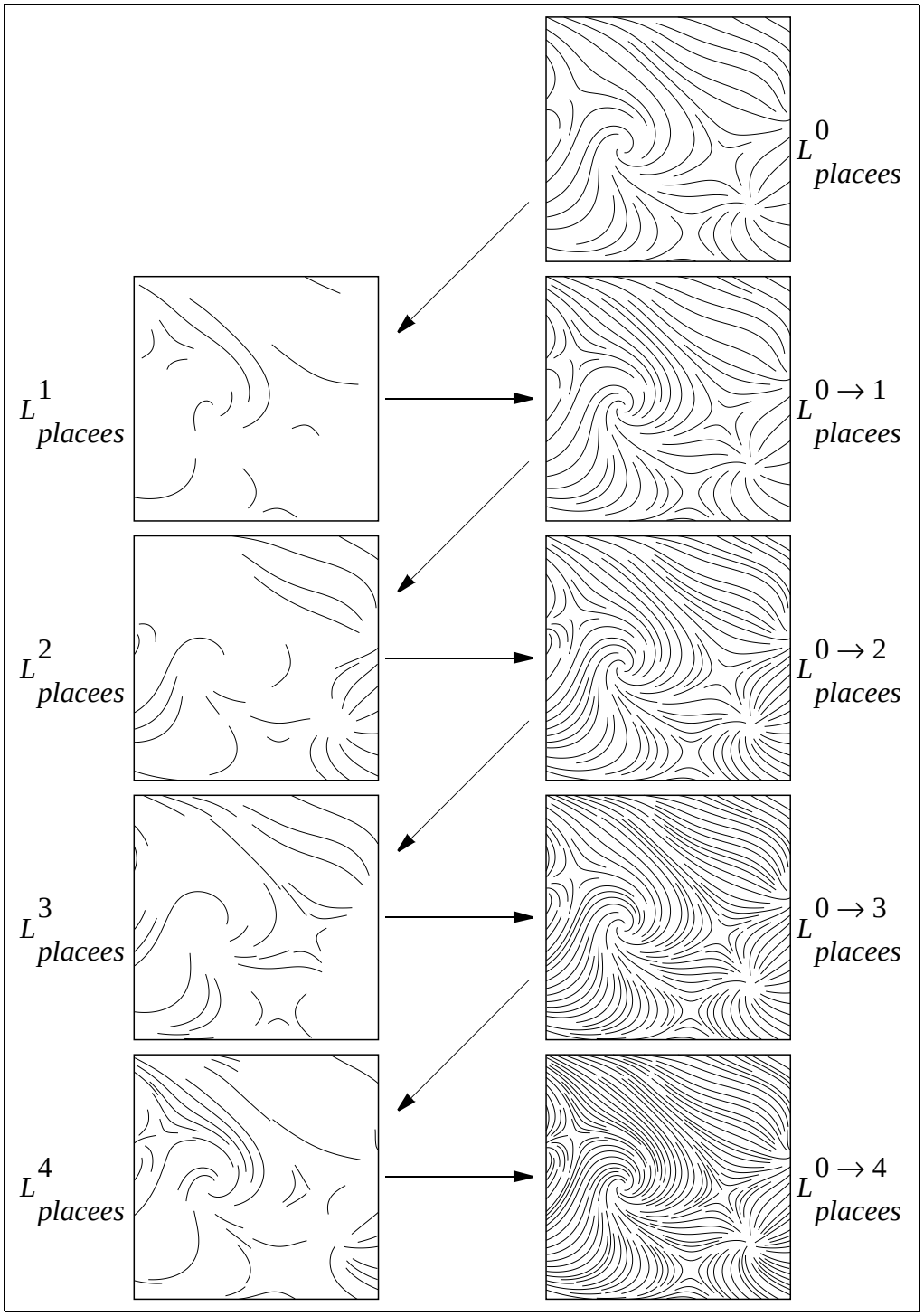

Figure 3.18 **Ensemble de streamlines multi-résolution**. Augmentation de la densité d'une représentation par l'ajout de streamlines construites pour des distances de séparation décroissantes. A gauche : ensembles de streamlines placées aux différents niveaux. A droite : ensembles de streamlines placées aux différentes densités.

4 Rendu visuel des streamlines

En jouant sur la distance de séparation entre les streamlines, notre algorithme de placement permet d'obtenir des représentations de densité quelconque. Nous avons étudié les techniques de rendu les plus efficaces pour les différentes densités, que nous présentons maintenant.

4.1 Représentations éparses

Nous appelons représentations éparses les représentations où les streamlines apparaissent distinctement les unes des autres. Les streamlines sont alors dessinées en tant que courbes dans le plan, l'épaisseur du trait pouvant varier en fonction de l'effet visuel souhaité.

4.1.1 Affinage des extrémités des streamlines (effet *tapering*)

Quand les streamlines sont dessinées avec une certaine épaisseur, leurs extrémités apparaissent comme des ruptures contrastées dans la représentation et perturbent son observation (voir figure 3.19a). De façon à minimiser cet artefact visuel, Turk et Banks ont introduit l'effet «*tapering*» qui consiste à effiler les extrémités des streamlines [55]. La conception de notre algorithme nous a permis d'inclure cette caractéristique d'affinage des extrémités des streamlines dans le processus de construction (voir figure 3.19b).

Pour ce faire, nous définissons l'épaisseur d'une streamline comme fonction de sa distance aux streamlines déjà placées. Ainsi si une streamline se rapproche d'une autre lors de sa construction, son épaisseur commence à décroître à partir d'une distance seuil que nous avons choisie égale à d_{seed} jusqu'à une épaisseur minimale lorsqu'elle atteint la distance de séparation d_{sep}. A cet instant la croissance de la streamline est stoppée dans cette direction. L'information d'épaisseur est affectée à chaque sample point.

Une passe de correction est cependant nécessaire quand la streamline a fini sa croissance car il est possible que des fluctuations d'épaisseur (augmentation et diminution) apparaissent le long de la streamline. Ceci se produit lorsqu'une streamline se rapproche d'une autre (son épaisseur diminue alors) puis s'en éloigne (son épaisseur croît à nouveau). Ces artefacts sont supprimés en parcourant les sample points de la streamline à partir d'une de ses extrémités et en affectant à chacun une épaisseur égale au maximum entre sa propre valeur et celle du sample point précédent. Quand l'épaisseur maximale est atteinte (à un sample point dit d'arrêt), le parcours est recommencé à partir de l'autre extrémité de la streamline en affectant de la même manière des épaisseurs croissantes jusqu'à atteindre le sample point d'arrêt. Cette correction rapide permet d'améliorer de façon significative la qualité des images.

Dans l'algorithme de Turk et Banks, l'effet *tapering* constitue une phase supplémentaire d'optimisation de l'algorithme. Une fois les streamlines placées, le processus de minimisation du niveau d'énergie de l'image filtrée est à nouveau exécuté en n'autorisant comme opération que l'affinage des extrémités des streamlines. Le processus est stoppé quand l'utilisateur estime que l'optimisation a atteint un niveau suffisant. L'avantage de notre approche est d'inclure le traitement de l'épaisseur dans la même passe que la construction des streamlines et de ne pas nécessiter d'intervention manuelle.

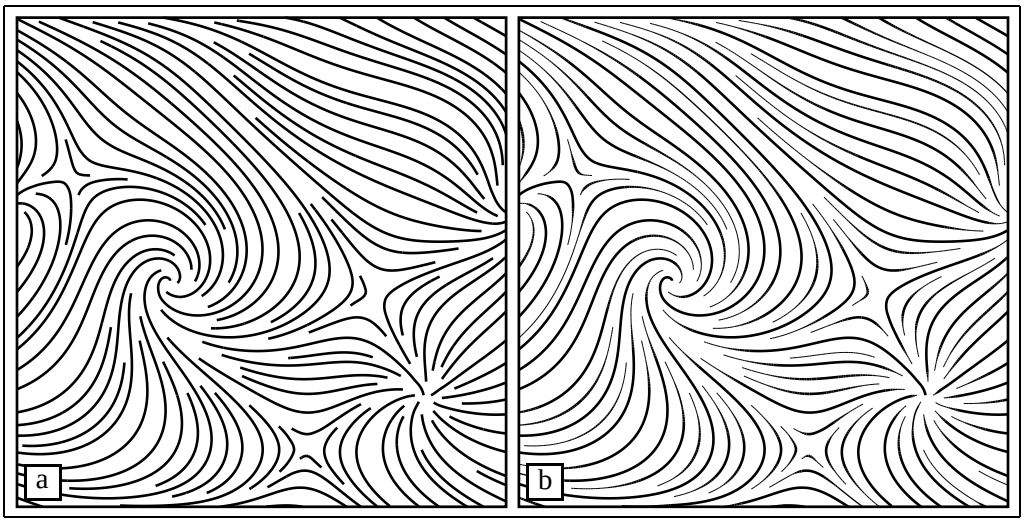

Figure 3.19 Effet *tapering*. (a) sans et (b) avec diminution de l'épaisseur des extrémités des streamlines.

4.1.2 Streamlines comme supports à des icônes

Une fois placées, les streamlines peuvent être considérées comme des squelettes qu'il est possible d'habiller avec des formes plus complexes. Si ces formes sont orientées il est alors possible de visualiser l'orientation du flux en plus de la direction. Les icônes placées sur de longues streamlines sont bien alignées et permettent une interprétation aisée de la direction et de l'orientation du flux, contrairement aux images traditionnelles à base d'icônes qui ne font pas apparaître explicitement les trajectoires dans le flux. Le placement des icônes peut être réalisé en mode vectoriel ou bitmap en plaquant de petites textures 2D.

Mode vectoriel

En mode vectoriel, les icônes sont composées par un ensemble de primitives graphiques (segment de droite, cercle, ...). Si on définit l'icône par un polygone, la position de ses sommets peut être déterminée en fonction de la position de sample points consécutifs sur une streamline (voir figure 3.20).

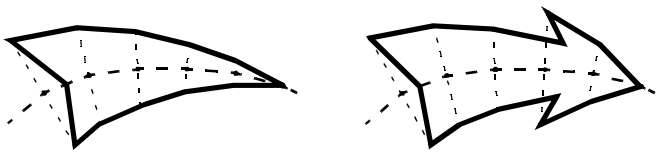

Figure 3.20 Le contour des icônes est calculé en fonction de la position des sample points d'une streamline.

L'avantage de cette représentation est que l'ensemble des icônes peut être agrandi ou réduit sans perte de qualité. Le mode vectoriel est particulièrement adapté à l'édition papier car il permet de tirer parti de la haute résolution des imprimantes. Un format de fichier couramment utilisé pour le stockage de graphiques vectoriels est Postscript. La figure 3.21 montre des représentations dans lesquelles les streamlines ont servi de support à des icônes triangulaires.

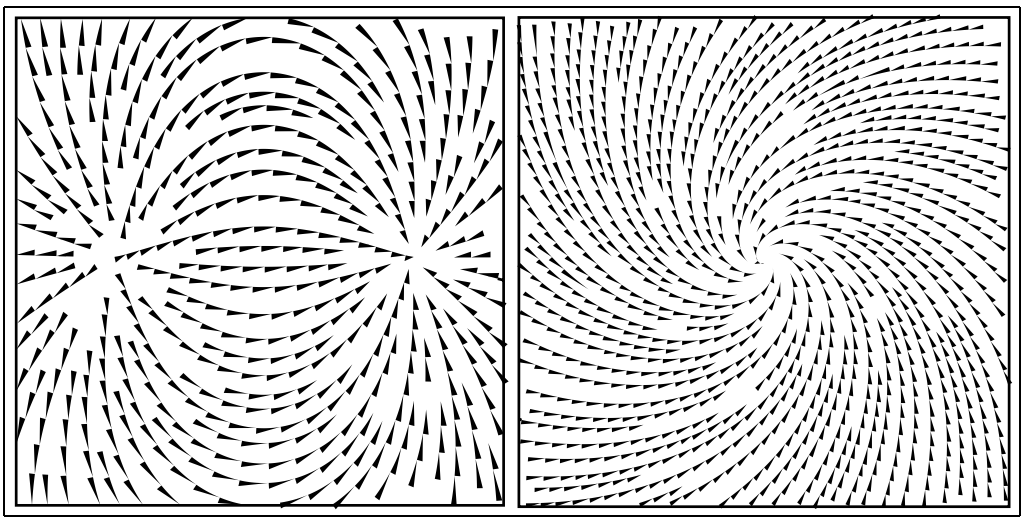

Figure 3.21 L'orientation du flux est révélée par le plaquage d'icônes orientées sur les streamlines

Mode bitmap

Une alternative au dessin des contours d'une icône est de plaquer de petites textures 2D le long des streamlines. La technique consiste à placer des polygones sur les streamlines et à les déformer selon la courbure de la streamline. La texture représentant l'icône est ensuite plaquée dans le polygone déformé (voir figure 3.22). L'avantage de ce mode est qu'il permet d'utiliser n'importe quelle image comme icône.

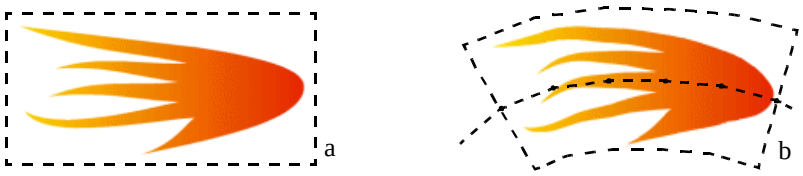

Figure 3.22 L'image de l'icône (a) est plaquée dans un polygone déformé selon la streamline (b).

4.1.3 Coloration des streamlines

La couleur sur les streamlines peut être utilisée pour afficher des informations supplémentaires. Ainsi il est possible de visualiser l'orientation du flux en plaquant des dégradés de couleurs orientés sur les streamlines. L'affectation des couleurs se fait sur le même principe que pour les représentations denses, abordées dans la section suivante.

4.2 Représentations denses

Les représentations dense de champs de vecteurs sont intéressantes pour leur résolution spatiale élevée. Elles permettent de connaître la direction du flux en n'importe quel point du domaine sans avoir à l'interpoler mentalement à partir d'éléments clairsemés, ce qui est particulièrement utile pour les champs de vecteurs turbulents.

En choisissant une distance de séparation de plus en plus petite, les streamlines se rapprochent jusqu'à ce qu'on ne puisse plus les distinguer. Il devient alors nécessaire de les différencier afin d'apprécier les directions du flux. Dans des techniques de visualisation de flux comme *LIC* ou *SpotNoise*, qui produisent également des représentations denses, l'intensité des pixels varie le long des lignes de courant. Le fait que ces intensités soient corrélées entre elles suivant ces lignes et pas du tout perpendiculairement permet d'interpréter les directions du flux.

Nous avons repris ce concept pour différencier nos streamlines. Des dégradés de niveau de gris leur sont superposés. Ces dégradés peuvent être appliqués sur les streamlines de différentes manières : soit en considérant une streamline comme une suite de petits segments colorés, soit en plaquant sur chacune d'elle des textures 1D. Ces deux approches sont exposées dans les sections suivantes.

Une caractéristique très intéressante de l'utilisation de dégradés sur les textures est le fait qu'il suffit d'utiliser des dégradés orientés pour visualiser l'orientation du flux. Rappelons que *SpotNoise* ne propose pas à ce jour la représentation de l'orientation sur des images statiques et qu'il a fallu attendre l'utilisation de filtres asymétriques [57] pour que *LIC* le permette.

Segments de couleur

L'idée de décomposer la streamline en segments de couleur a déjà été exploitée dans les articles de van Gelder [21] et de Max [44]. Techniquement la mise en oeuvre est simple. Il suffit d'allouer une couleur à chaque sample point, la streamline est alors tracée en affichant les segments reliant deux sample points consécutifs avec la couleur du premier par exemple. L'important de la technique repose sur le mode d'allocation des couleurs aux sample points de la streamline.

Dans le cas de dégradés, il faut allouer à des sample points consécutifs des intensités croissantes ou décroissantes. Une méthode consiste à utiliser une fonction d'intensité *i(x)* qui donne au sample point de rang *x* son intensité correspondante. La courbe des intensités en fonction de leur rang détermine l'allure des motifs sur la streamline. A titre d'exemple, considérons les deux fonctions périodiques suivantes :

$$i_1(x) = \frac{1}{2}\left(1 + sin\left(\frac{2\pi x}{N}\right)\right) \quad \text{et} \quad i_2(x) = \frac{x \bmod N}{N-1}$$

où *N* est la période de la fonction d'intensité (mesurée en nombre de sample points). Plus la période N est longue et plus les motifs sur les streamlines apparaîtront étirés. La notion de longueur de période des fonctions n'a véritablement un sens sur la représentation finale que si la distance entre les sample points consécutifs des streamlines est constante. Dans ce cas la longueur des motifs sur les streamlines sera constante dans toutes les régions du domaine.

i_1, qui est une fonction sinusoïdale, produit des «vagues» continues d'intensités croissantes et décroissantes, alors que i_2, qui comporte un modulo, donne des segments discontinus d'intensités croissantes qui lèvent l'ambiguïté sur l'orientation du flux. Les figures 3.23 et 3.24 montrent les images obtenues avec ces deux fonctions pour des périodes différentes.

On remarque que la longueur de la période des fonctions a une influence sur la facilité d'interprétation du flux. Ainsi les images produites avec de longues périodes (figure 3.23b et 3.24b) semblent mieux montrer la structure des champs de vecteurs. Notre méthode d'allocation permet facilement de jouer sur l'apparence de la représentation finale en choisissant une fonction d'intensité adaptée, et cela sans surcoût de calcul.

Remarquons enfin que les images obtenues avec une fonction asymétrique sont très similaires à celles obtenues par la technique de génération de textures synthétiques développée par Khouras [35], mais notre méthode est beaucoup plus simple à contrôler puisqu'il suffit de définir la fonction d'intensité.

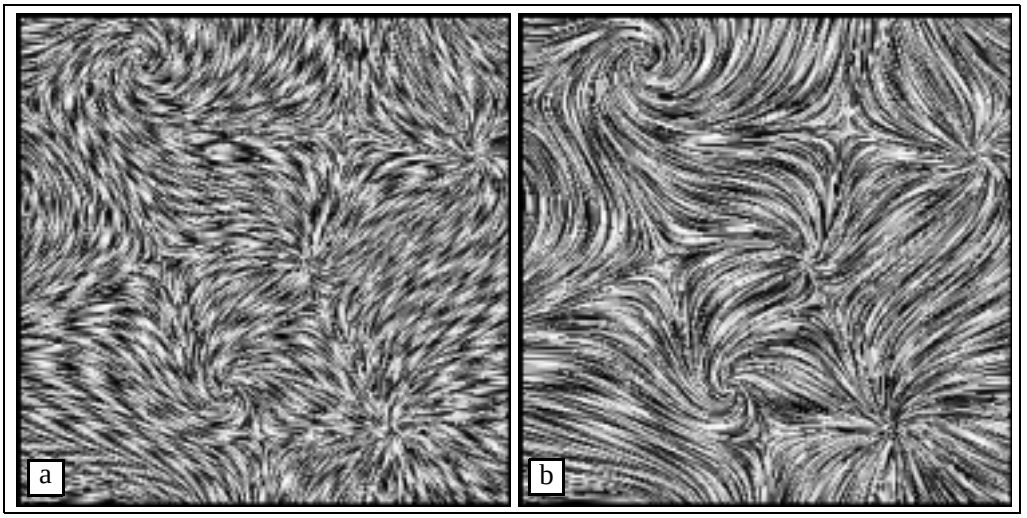

Figure 3.23 Utilisation de la fonction sinusoïdale i_1 avec une période (a) courte et (b) longue.

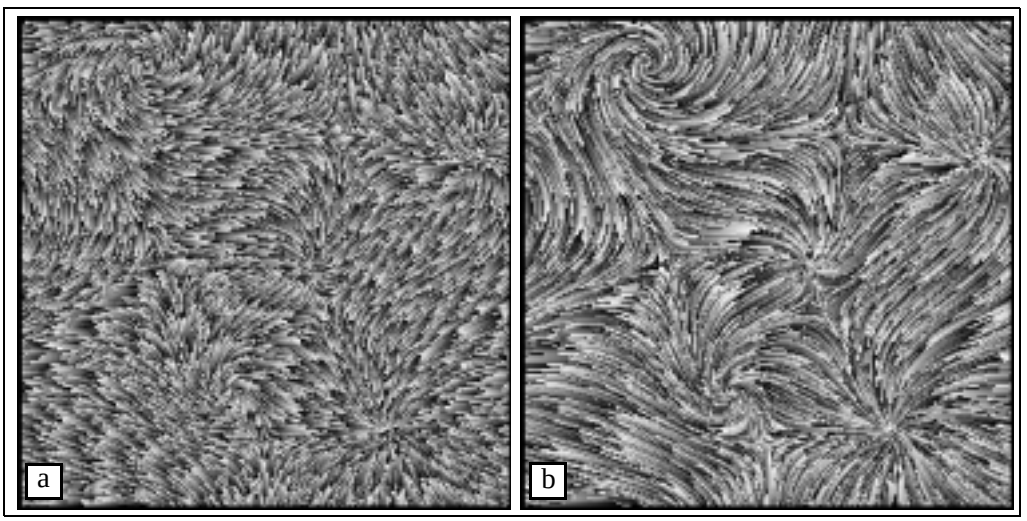

Figure 3.24 Utilisation de la fonction asymétrique i_2 avec une période (a) courte et (b) longue.

Textures 1D

Il est possible d'obtenir des dégradés sur les streamlines en plaquant des textures mono-dimensionnelles. Dans ce cas, les intensités du dégradé ne sont pas directement affectées aux sample points de la streamline mais aux pixels constituant la texture 1D. La texture est ensuite plaquée répétitivement sur chaque streamline. Le choix d'une texture asymétrique permet de visualiser l'orientation du flux.

Le mécanisme de plaquage de texture nécessite d'allouer à chaque sample point des coordonnées de texture croissantes. L'incrément de la coordonnée de texture entre deux sample points consécutifs est $1/N$ si N représente la longueur en sample points du support de la texture. Plus N est grand et plus la texture est étirée sur un grand nombre de sample points. Ce paramètre N joue le même rôle que la période des fonctions d'intensité utilisées dans le paragraphe précédent.

L'avantage de l'utilisation de texture 1D est principalement visuel. En effet le plaquage de texture se charge d'interpoler les couleurs entre deux sample points consécutifs, donnant ainsi un meilleur rendu qu'avec des segments de couleur constante.

5 Résultats

Nous comparons les représentations obtenues avec notre algorithme avec celles calculées par la méthode de Turk et Banks, qui est connue comme étant le meilleur algorithme de placement existant à ce jour dans la littérature [55]. Nous pouvons remarquer sur la figure 3.25 que les représentations obtenues pour les mêmes densités sont semblables mais nous voyons dans le tableau 2 que les temps de calculs pour notre méthode sont nettement inférieurs. Les bons résultats de notre algorithme s'expliquent par le choix d'un *critère absolu* pour le placement des streamlines comparativement au *critère relatif* utilisé par Turk et Banks. Dans leur méthode, un niveau d'énergie est affectée à une distribution de streamlines, puis de meilleures distributions (ayant un niveau d'énergie inférieur) sont recherchées en remplaçant aléatoirement des streamlines par d'autres. Ce processus d'optimisation requiert l'intégration de beaucoup plus de streamlines que celles présentes dans la représentation finale, c'est ce surplus de calcul qui pénalise l'approche. Dans notre algorithme, en prenant la distance de séparation comme critère de placement, le placement des streamlines est *direct*. Chaque streamline est placée définitivement dans le domaine et n'est jamais remise en cause. Le nombre d'intégrations numériques est donc réduit au minimum.

Distance de séparation en % de la largeur du domaine	**Image-guided streamline placement** (Turk-Banks)	**Placement direct** (notre approche)
6%	Fig 3.25a : stoppé à 2 mn	Fig 3.25d : 4 secondes
3%	Fig 3.25b : stoppé à 4 mn	Fig 3.25e : 9 secondes
1.5%	Fig 3.25c : stoppé à 10 mn	Fig 3.25f : 17 secondes

Tableau 2: Comparaison des temps de calcul obtenus sur un processeur MIPS R4600PC à 100MHz. Les images et les temps de calcul pour l'image-guided placement ont été obtenus avec le programme original de G. Turk.

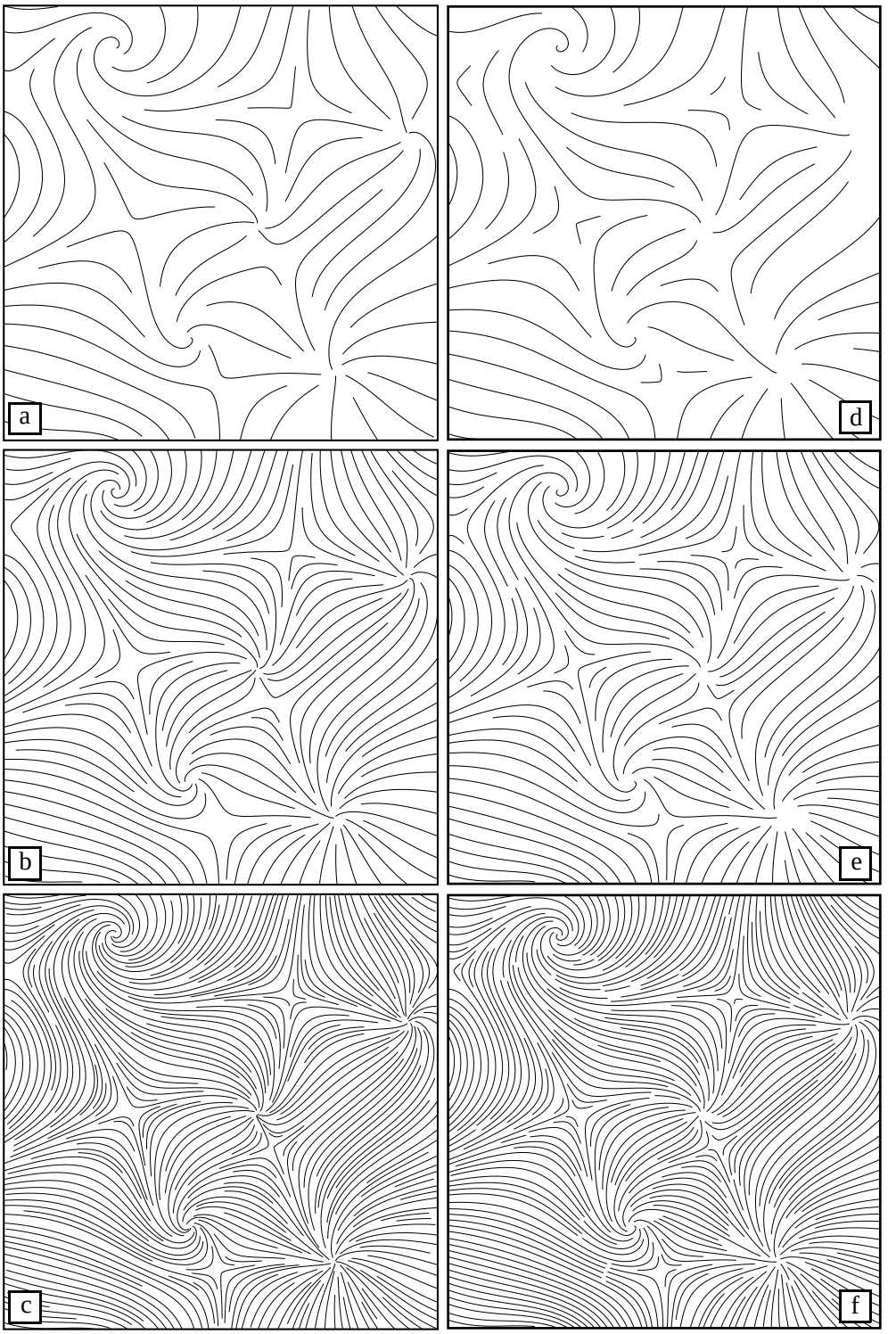

Figure 3.25 Comparaison des représentations obtenues pour des distances de séparation de 6% (a, d), 3% (b, e) et 1.5% (c, f) de la largeur du domaine. Colonne de gauche : *Image-guided streamline placement* de Turk et Banks. Colonne de droite : notre algorithme de placement.

5.1 Streamlines multi-résolution

Nous avons développé un module de navigation permettant l'exploration interactive d'un ensemble de streamlines multirésolution. La figure 3.26 montre quelques capture d'écran de notre système de navigation.

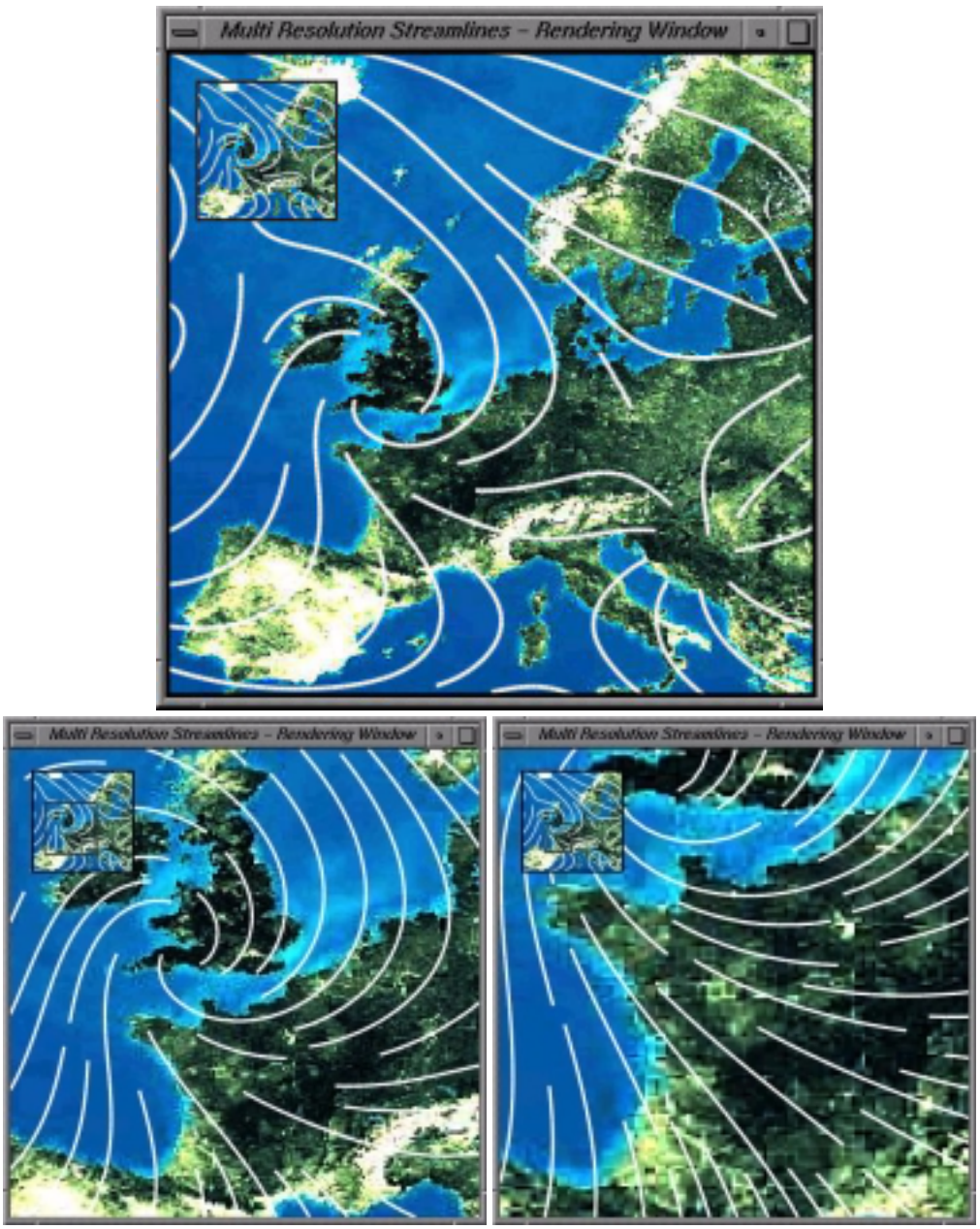

Figure 3.26 Navigation dans un ensemble de streamlines multi-résolution. La densité de streamlines dans l'image est gardée constante à mesure que l'on zoome dans le domaine. Une fenêtre de supervision indique la région visualisée.

6 Conclusion

Nous avons présenté dans ce chapitre un algorithme de placement de streamlines en 2D permettant un contrôle accru sur la densité des représentations obtenues. Le contrôle de la densité est effectué en fixant la distance de séparation entre les streamlines. En faisant varier ce paramètre, les représentations possibles vont de éparse à dense.

La mise en oeuvre de plusieurs méthodes de sélection de seed points nous a permis de trouver une technique de sélection originale et efficace en initiant des streamlines dans le voisinage des streamlines déjà placées.

La méthode de placement a pu être étendue pour obtenir des ensembles de streamlines multi-résolutions permettant de visualiser un champ de vecteurs à plusieurs niveaux de détails. Un système de navigation interactive utilisant une telle représentation a été développé permettant l'exploration interactive des champs de vecteurs.

Nous avons proposé des modes de rendu visuel permettant de tirer le meilleur profit des distributions éparses et denses de streamlines.

Nos résultats ont été comparés à la technique de placement de streamlines de Turk et Banks [55], reconnue comme étant la meilleure à ce jour. Nous avons montré que les temps de calcul de notre méthode sont très inférieurs (d'un facteur 30 environ) pour une qualité visuelle identique.

Une partie de ces travaux ont été présentés à la conférence Eurographics Workshop on Visualization in Scientific Computing [30]. Cette publication à donné lieu à une extension de notre algorithme au placement de streamlines en 3D par Fuhrmann et Gröller [20].

CHAPITRE

4

La *Motion Map* : Animation de champs de vecteurs stationnaires

1 Introduction

De nombreuses recherches ont été menées depuis quelques années dans le domaine de la visualisation des champs de vecteurs stationnaires pour l'obtention de représentions denses permettant l'interprétation des caractéristiques du flux en tout point du domaine. *LIC* et *SpotNoise* sont les approches les plus connues issues de ces recherches. Ces deux techniques, sous beaucoup d'aspects comparables [15], obtiennent une bonne corrélation spatiale des pixels, permettant de visualiser la structure du champ de vecteurs dans une image statique. Elles ont été toutes les deux étendues afin de produire des animations révélant l'orientation du flux. La corrélation spatiale de ces deux techniques est basée sur le calcul de petites streamlines qui servent de support au noyau de convolution pour LIC et à la projection des spots pour SpotNoise. Afin de générer des représentations denses, ces deux techniques doivent calculer suffisamment de streamlines pour couvrir la totalité des pixels et cela pour chaque image de l'animation, l'intégration des streamlines étant la partie coûteuse de ces méthodes.

Une propriété d'un champ de vecteurs stationnaire étant d'avoir sa structure invariante au cours du temps, on peut s'interroger sur la nécessité d'avoir à recalculer un ensemble de streamlines à chaque pas de temps de l'animation. Nous sommes partis de cette réflexion pour ne calculer qu'une seule fois cet ensemble, l'information nécessaire à l'apparition de mouvement étant ensuite ajouté sur cette structure fixe. Le principe de notre approche consiste à calculer un ensemble dense de streamlines couvrant le domaine puis d'y allouer des couleurs de manière à introduire un effet de mouvement par le recours à la technique d'animation de la palette de couleurs [50]. L'ensemble de ces informations, structure plus information de mouvement, vont pouvoir être

stockées dans une structure de donnée unique pour toute l'animation que nous avons appelée *Motion Map*.

Nous avons vu dans le chapitre précédent que notre algorithme de placement de streamlines permettait de produire des représentations denses de champs de vecteurs en choisissant une distance de séparation suffisamment petite pour que les streamlines adjacentes se touchent. Bien que la conception de l'algorithme de placement lui permette d'obtenir des distributions de streamlines de densité quelconque, dense en l'occurrence, nous l'avons adapté pour ce cas particulier afin d'en améliorer l'efficacité. Pour distinguer les streamlines les unes des autres, nous avions plaqué dessus des motifs de type dégradé de couleurs à l'aide de *fonctions d'intensité*. L'orientation du flux apparaissait alors sur une image statique en utilisant des dégradés orientés. Afin de déplacer les motifs formés par les dégradés dans le cadre d'une animation, nous avons développé une autre technique de coloration des streamlines.

Finalement, l'adaptation de notre algorithme de placement associé à notre technique d'allocation des couleurs sur les streamlines nous permet de produire des animations de champs de vecteurs stationnaires ayant les caractéristiques suivantes :

- animation parfaitement cyclique et représentation de vitesses variables sur les streamlines,
- animation en temps réel par la technique d'animation de la table des couleurs ou en utilisant l'accélération matériel du plaquage de texture,
- rapidité de calcul : le calcul d'une animation complète est à peine plus coûteux que celui d'une seule image,
- faible encombrement mémoire : toute l'information nécessaire à l'animation est contenue dans une seule image.

Nous décrivons dans la section 2 le mode d'allocation des couleurs sur les streamlines qui utilise la technique d'animation de la palette des couleurs puis en section 3 nous détaillons la façon dont toute l'information nécessaire à l'animation de flux stationnaires est stockée dans une structure de donnée originale appelée Motion Map. La section 4 propose deux moyens de visualiser les animations créées et la section 5 présente des résultats et commentaires sur la méthode avant de conclure par la section 6.

2 Codage du mouvement sur les streamlines

La coloration des streamlines a deux objectifs principaux. D'une part elle permet de différencier les streamlines les unes des autres quand leur distribution dans le domaine devient dense, en corrélant entre eux les pixels appartenant à une même streamline. D'autre part, si les couleurs appliquées aux streamlines engendrent des motifs orientés, cela permet de visualiser l'orientation du flux.

Dans cette section, nous allons montrer comment il est possible de rendre l'interprétation du flux plus directe et intuitive en animant des motifs sur les streamlines plutôt qu'en choisissant des motifs orientés. Le mode d'allocation des couleurs que nous avons développé est basé sur la technique d'*animation de la palette des couleurs*. Nous commencerons d'abord par décrire les caractéristiques de cette technique avant d'en adapter les principes à l'allocation des couleurs sur les streamlines dans la section 2.2.

2.1 Animation de la palette des couleurs

L'animation de la palette des couleurs, introduite par Shoup [50], permet d'obtenir une animation à partir d'une seule image. Cette technique tire parti du mode fausses-couleurs des cartes graphiques dans lesquelles les couleurs sont regroupées dans un tableau (la palette des couleurs) et dont un indice est alloué à chaque pixel de l'image. Lors de l'affichage de l'image en mode fausses-couleurs, la couleur de chaque pixel est déterminé par son indice dans la palette des couleurs. Avec ce mode il est possible de changer l'aspect de l'image à l'écran en modifiant seulement les couleurs dans la palette, sans changer les pixels de l'image.

La technique d'animation de la palette des couleurs utilise cette caractéristique de manière à introduire une impression de mouvement dans l'image en modifiant successivement le contenu de la palette. L'opération classique consiste en un décalage circulaire d'un sous ensemble de couleurs de la palette. Ainsi si ce sous ensemble contient N couleurs, l'image passe par N aspects différents lors de la rotation des couleurs. L'animation ainsi produite est cyclique et contient N images distinctes. Comme la structure de l'image est invariante au cours de l'animation, les indices des pixels doivent être judicieusement répartis dans l'image de façon à ce que la rotation des couleurs dans la palette se traduise par le déplacement de motifs plutôt qu'un simple clignotement des pixels.

L'intérêt principal de cette technique est qu'elle permet de coder une animation en ne définissant qu'une image et une palette de couleur associée. L'animation est rapide et peut être faite en temps réel, même sur des ordinateurs bas de gamme.

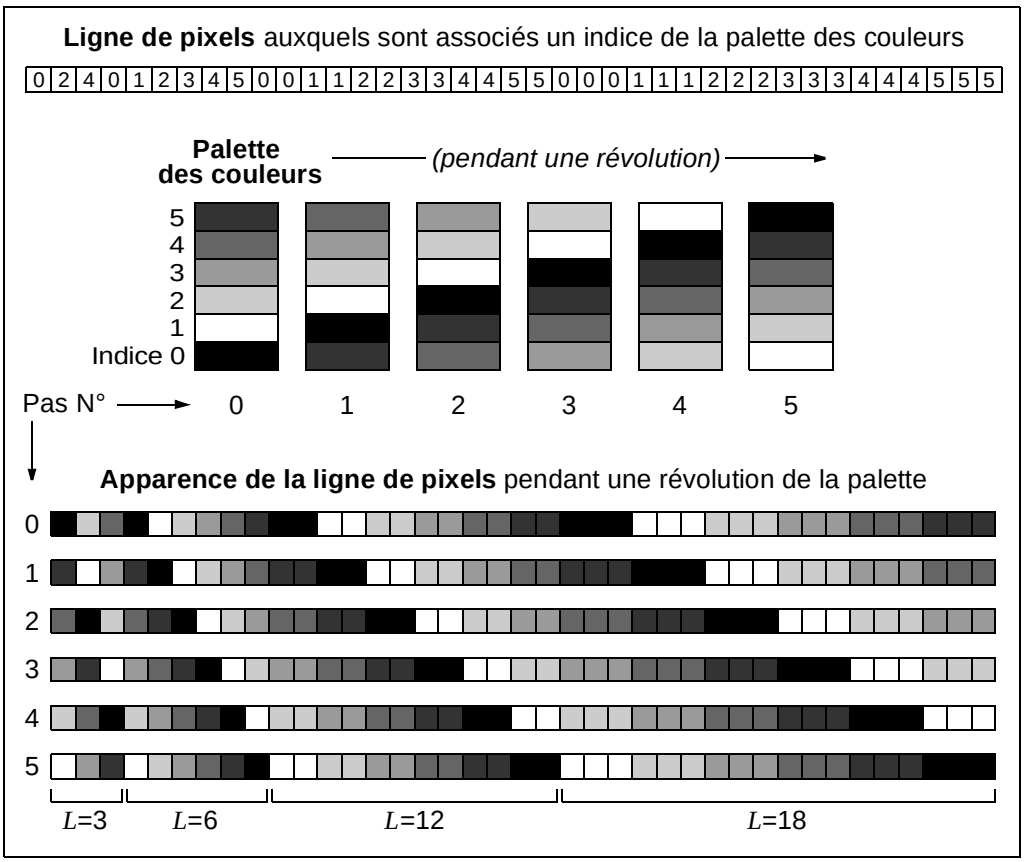

Figure 4.1 **animation de la palette des couleurs**. Les indices d'une table de N couleurs sont associés à des pixels (ici N=6). Le contenu de la palette est décalé circulairement N fois (représentant un cycle complet de l'animation). L'apparence successive de la ligne de pixels donne l'impression de déplacement des motifs sur la ligne de pixels.

Nous n'allons pas nous intéresser à toutes les possibilités qu'offre cette technique d'animation mais seulement aux aspects transposables ultérieurement à l'animation de motifs sur des streamlines, c'est à dire une animation sur un support monodimensionnel. Nous décrivons dans la suite de cette section les concepts utiles à partir d'un exemple montrant comment la répartition des indices d'une palette de couleurs sur une ligne de pixels permet d'obtenir un effet de mouvement lors de la rotation de la palette. Cette exemple est illustré par la figure 4.1.

On remarque sur cette figure qu'il a suffit d'allouer incrémentalement et périodiquement les indices de la palette aux pixels consécutifs pour qu'un déplacement des motifs se produise. Notons que l'allocation incrémentale des indices sur la ligne de pixels préserve l'allure du motif défini par l'agencement des couleurs dans la palette (ici, un dégradé de gris). La répartition des indices Ind_{p_i} sur les pixels p_i de la ligne est donnée par une formule récurrente de la forme :

$$\begin{cases} I_{p_0} = 0 \\ I_{p_i} = I_{p_{i-1}} + k_i \\ Ind_{p_i} = [I_{p_i}] \ mod \ N \end{cases} \tag{4.1}$$

où (I_{p_i}) est une suite croissante de valeurs réelles avec $i \in \{0, \ldots, n_p - 1\}$ (n_p est le nombre de pixels sur la ligne), k_i est un incrément réel positif, $[I_{p_i}]$ est la partie entière de I_{p_i}, N est le nombre de *couleurs animées* dans la palette et *mod* est la fonction modulo.

La vitesse apparente du déplacement des motifs sur la ligne de pixels est déterminée par l'incrément k_i dans l'équation (4.1). Une vitesse constante de déplacement est obtenue avec un incrément k_i constant. La variation de cet incrément au cours de l'allocation des indices aux pixels permet d'obtenir la représentation de vitesses variables sur une même ligne de pixels.

La répartition des $N = 6$ indices sur les $n_p = 39$ pixels p_i de la figure 4.1 a été obtenue à partir de la formule (4.1) avec un incrément k_i égal à 2 pour $i \in \{1, 2, 3\}$, 1 pour $i \in \{4, \ldots, 9\}$, 1/2 pour $i \in \{10, \ldots, 21\}$ et 1/3 pour $i \in \{22, \ldots, 38\}$. La valeur décroissante de l'incrément k_i le long de la ligne de pixels a pour effet de faire croître la vitesse de déplacement des motifs. L'exemple a été construit de telle manière que la vitesse soit constante par morceaux et que la longueur des segments de pixels corresponde à la longueur nécessaire pour plaquer un cycle de couleurs de la palette. On remarque que la longueur L des segments de vitesse constante est donnée par

$$L = \frac{N}{k_i} \tag{4.2}$$

Sur l'exemple, la longueur minimum L_{min} est 3 ($N/k_1 = 6/2$) et la longueur maximum L_{max} est 18 ($N/k_{22} = 6/(1/3)$).

C'est ce mode d'allocation incrémentale, d'un pixel à l'autre, qui permet d'introduire une corrélation spatiale forte entre les pixels de la ligne sous forme d'une relation de succession. Lors de la rotation des couleurs dans la palette, ces couleurs semblent suivre la ligne puisqu'elles se retrouvent successivement affichées sur des pixels voisins. Une bonne corrélation spatiale garantit alors une bonne corrélation temporelle. Notons encore que plus les supports spatiaux sont longs (ici la ligne de pixels) et plus les corrélations spatiales et temporelles dans l'image seront bonnes.

C'est ce mécanisme de corrélation des pixels que nous allons exploiter pour allouer des indices de couleur sur les streamlines qui sont aussi des supports monodimensionnels. La rotation des couleurs dans la palette sera alors suffisante à l'apparition d'un mouvement sur les streamlines.

2.2 Allocation des indices de couleurs sur les streamlines

La technique d'allocation des indices de couleurs sur une ligne de pixels est très facilement transposable au cas des streamlines. Nous utilisons pour cela une fonction qui donne l'incrément *k* en fonction de la vitesse locale du flux. Les indices de couleurs ne sont plus directement affectés à des pixels mais aux sample points de chaque streamline. C'est seulement lorsque les streamlines seront discrétisées dans une image ou à l'écran que les informations de couleur seront affectées aux pixels. Ce dernier aspect est abordé dans la section 3.3.

L'allocation des indices de couleurs aux sample points des streamlines est inspirée de l'équation (4.1), elle est décrite par le système suivant :

$$\begin{cases} I_{p_0} = random(N) \\ I_{p_i} = I_{p_{i-1}} + k(V_{p_{i-1}}) \\ Ind_{p_i} = [I_{p_i}] \ mod \ N \end{cases} \tag{4.3}$$

où les p_i représentent maintenant des sample points et $k(V_{p_{i-1}})$ est l'incrément positif calculé en fonction de la vitesse locale au niveau du sample point précédent. La fonction *random*(*N*) associe un indice de couleur aléatoire au premier sample point pour que les streamlines adjacentes soient différenciables et pour éviter de former des motifs non porteurs d'information. La figure 4.2 montre deux représentations du même champ de vecteurs obtenues avec une initialisation de I_{p_0} à 0 et à *random*(*N*), les motifs qui apparaissent sur la figure 4.2a sont des artefact de visualisation.

L'allocation incrémentale des indices sur les sample points revient à plaquer périodiquement les couleurs de la palette sur la streamline. Comme pour les fonctions d'intensité *i(x)* introduites dans la section 4.2 du chapitre précédent, il est intéressant de pouvoir jouer sur l'aspect visuel de la représentation en contrôlant la longueur de la période de la fonction. Dans le cas des fonctions d'intensité, la période était unique puisqu'elle ne dépendait pas de la vitesse du flux. Comme nous souhaitons visualiser des vitesses variables sur une même streamline, il nous faut maintenant préciser les bornes de l'intervalle des périodes accessibles en fixant leurs longueurs minimum L_{min} et maximum L_{max} (mesurées en sample points) en fonction des vitesses locales minimum V_{min} et maximum V_{max} du champ de vecteurs. L'intérêt de préciser les valeurs L_{min} et L_{max} de façon à faire apparaître les disparités des vitesses locales dans le flux est illustré par la figure 4.3.

Rappelons que les vitesses locales du flux sont obtenues par interpolation linéaire des vecteurs disponibles dans le champ de vecteurs. Tout vecteur interpolé a donc une norme inférieure ou égale aux vecteurs qui ont servi a cette interpolation, on en déduit donc que la vitesse locale maximum V_{max} n'est autre que le vecteur ayant la norme maximum dans le champ. Pour la vitesse

minimum on ne peut plus se fier aux vecteurs du champ puisque même si la norme minimum est $v_{min} > 0$ il est possible que des points critiques (où la vitesse locale est nulle) se trouvent entre les données échantillonnées. Une heuristique pour le choix de V_{min} consiste à fixer sa valeur comme une fraction de la norme minimum des vecteurs v_{min} dans le champ et de considérer la vitesse v en tout point du champ comme étant le maximum entre la vitesse réelle et V_{min} ($v = max(v, V_{min})$). Il est aussi possible, si l'on sait que le champ de vecteurs contient des points critiques, de fixer la vitesse minimum locale V_{min} à 0.

De la formule (4.2) on déduit l'intervalle des valeurs que peut prendre la fonction d'incrément k en fonction de L_{min} et L_{max}. On peut ainsi écrire

$$\begin{cases} k(V_{min}) = N/L_{min} \\ k(V_{max}) = N/L_{max} \end{cases} \quad \text{avec} \quad \frac{N}{L_{max}} \leq k(v) \leq \frac{N}{L_{min}} \tag{4.4}$$

où $k(v)$ est l'incrément correspondant à une vitesse locale quelconque v comprise entre V_{min} et V_{max}. Cet incrément étant déterminé par une interpolation linéaire entre ses deux bornes, la fonction d'incrément k peut alors être écrite de la façon suivante :

$$k(v) = \frac{N}{L_{max}}\left(\frac{v - V_{min}}{V_{max} - V_{min}}\right) + \frac{N}{L_{min}}\left(\frac{V_{max} - v}{V_{max} - V_{min}}\right) \tag{4.5}$$

Dans la formulation de la fonction d'incrément de l'équation (4.5), les vitesses réelles du flux sont mises à l'échelle de façon à ce que les périodes d'indices de couleurs sur les streamlines prennent une valeur entre L_{min} et L_{max}. Ce recalage forcé ne traduit pas fidèlement les proportions réelles entre les vitesses locales du flux. Pour respecter le ratio R entre ces vitesses on fixe l'incrément $k(V_{max})$ et on déduit $k(V_{min})$ en posant :

$$\begin{cases} R = V_{min}/V_{max} \\ k(V_{max}) = N/L_{max} \\ k(V_{min}) = \dfrac{k(V_{max})}{R} = \dfrac{NV_{max}}{L_{max}V_{min}} \end{cases} \tag{4.6}$$

En pratique, nous avons observé que le mouvement est interprétable dans les régions de faible vitesse si L_{min} est supérieur ou égal à 3, ce qui implique d'après l'équation (4.4) que l'incrément pour la vitesse minimum $k(V_{min})$ doit être inférieur à $N/3$.

Si on pose $k(V_{min}) \leq \frac{N}{3}$, d'après (4.6) on obtient :

$$V_{min} \geq \frac{3V_{max}}{L_{max}} = V_{seuil} \tag{4.7}$$

La condition sur V_{min} nous montre que la visualisation du champ de vecteur en respectant le rapport entre les vitesses locales est de qualité optimale si aucune vitesse locale ne descend en dessous de la valeur seuil V_{seuil}. Pour visualiser un champ de vecteurs contenant des points critiques (pour lesquels la vitesse locale est nulle) on fixe $V_{min} \geq V_{seuil}$ et on ramène les vitesses inférieures à V_{min} à cette valeur durant le processus d'allocation des indices de couleur. Cette approximation permet de visualiser nettement le mouvement autour des points critiques. D'après l'équation (4.6) et sous la condition (4.7) l'expression de cette nouvelle fonction d'incrément s'écrit

$$k(v) = \frac{N}{L_{max}}\left(\frac{v - V_{min}}{V_{max} - V_{min}}\right) + \frac{NV_{max}}{L_{max}V_{min}}\left(\frac{V_{max} - v}{V_{max} - V_{min}}\right) \tag{4.8}$$

En résumé nous proposons deux fonctions d'incrément k pour l'allocation incrémentale des indices de couleur de la palette sur les sample points des streamlines. Le schéma de l'allocation incrémentale est donnée par l'équation (4.3). La première fonction d'incrément permet de spécifier explicitement l'*intervalle dynamique* des vitesses du flux en fournissant les valeurs L_{min} et L_{max} (voir figure 4.2), elle est donnée par la formule (4.5). La seconde fonction d'incrément permet de visualiser le mouvement du flux en respectant la proportion réelle de ses vitesses locales. Il suffit de fixer L_{max} et V_{min} selon la condition (4.7). Cette fonction d'incrément est donnée par la formule (4.8).

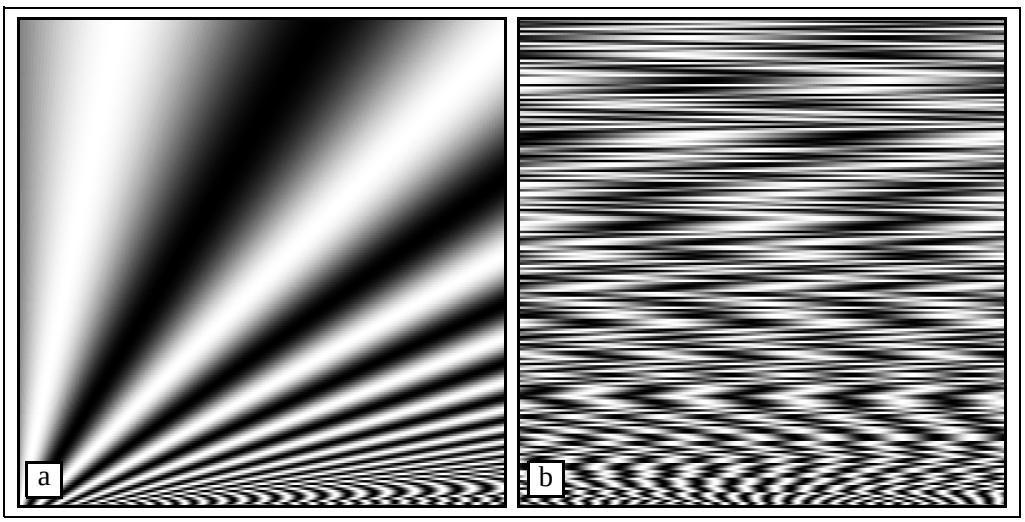

Figure 4.2 **Intervalle dynamique** sur un flux de vitesses croissantes. Pour visualiser l'éventail des vitesses de ce flux, nous avons fixé L_{min} et L_{max} à 3 et 100 respectivement. (gauche) $I_{p_0} = 0$, (droite) $I_{p_0} = random(N)$.

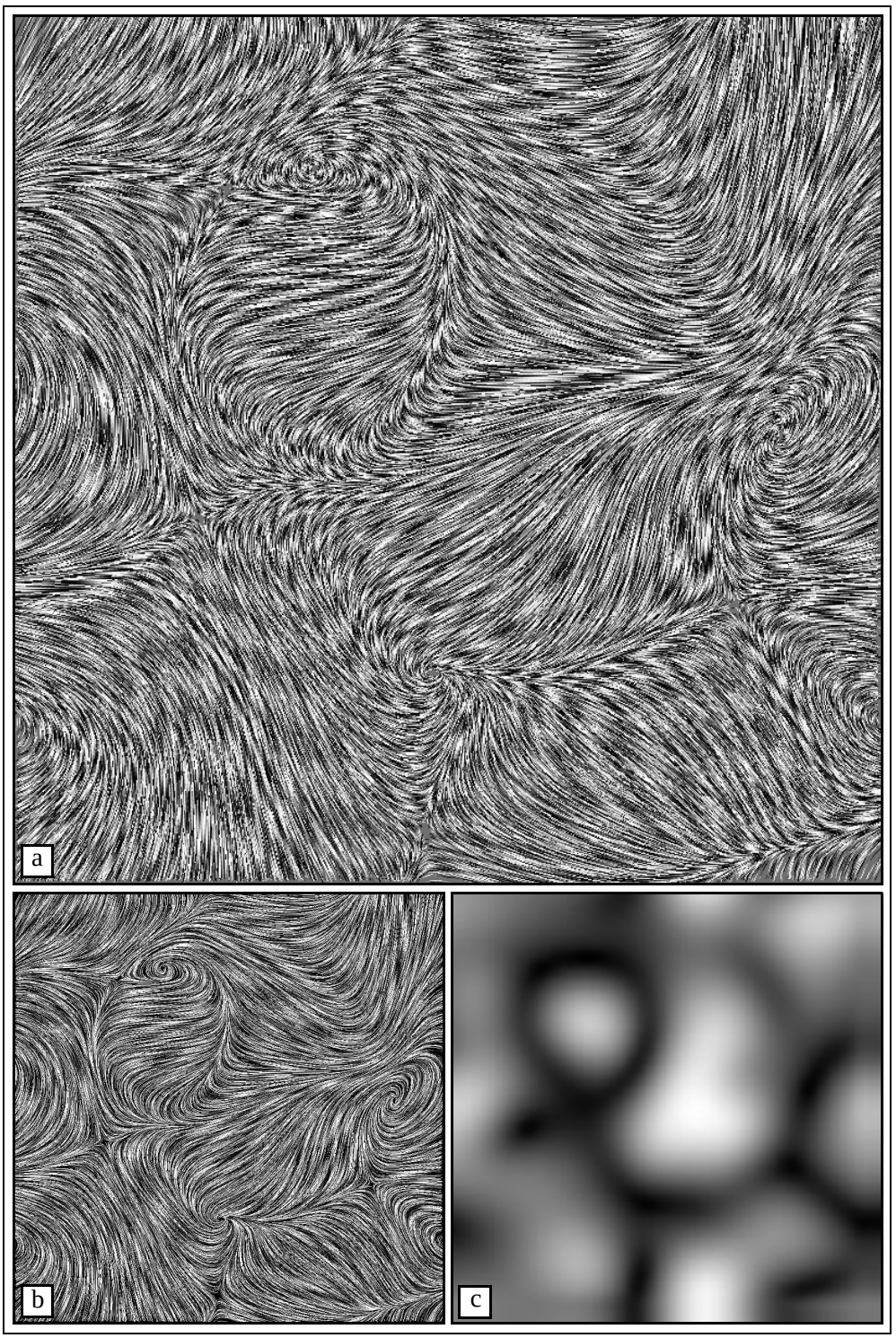

Figure 4.3 **Influence de L_{min} et L_{max}** sur la visualisation de la répartition des vitesses dans le flux. (b) $L_{min} = L_{max} = 30$, la vitesse du flux apparaît comme constante. (a) $L_{min} = 10$ et $L_{max} = 50$, les zones ayant des vitesses plus importantes apparaissent nettement. (c) carte des vitesses locales du flux.

L'intérêt de ce mode d'allocation des couleurs sur les streamlines est qu'il est maintenant possible, après avoir choisi une palette de couleurs et discrétisé les streamlines, de les animer en décalant les couleurs de la palette. Deux méthodes permettant de visualiser l'animation sont présentées dans la section 4. La section suivante concerne l'adaptation de notre algorithme de placement au cas des représentations denses de streamlines. Nous verrons que de telles représentations peuvent être stockées avec l'information de mouvement du flux dans une structure originale que nous avons appelé Motion Map.

3 La Motion Map

Nous avons vu dans les sections précédentes qu'il était possible d'allouer un indice de couleurs aux pixels traversés par une streamline de façon à obtenir une impression de mouvement par rotation de la palette de couleurs. En allouant des indices de couleurs sur un ensemble dense de streamlines nous pouvons alors visualiser la structure du flux en animant la totalité du domaine. Cette approche est valide, car rappelons le, la structure d'un champ de vecteurs stationnaire est invariante au cours du temps et peut être représentée par un seul ensemble de streamlines. Le grand intérêt de cette approche est qu'il suffit de conserver l'indice alloué à chaque pixel pour stocker la totalité de l'animation.

La Motion Map se présente sous la forme d'un tableau bidimensionnel dans lequel chaque cellule contient un indice de couleur. Ces indices ont été auparavant alloués aux sample points de l'ensemble dense de streamlines. C'est en discrétisant les streamlines dans la Motion Map que les indices de couleurs sont affectés à ses cellules. Il ne reste plus qu'à fournir une palette de couleur pour visualiser le contenu de la Motion Map et d'effectuer une rotation des couleurs dans la palette pour produire une animation. La figure 4.4 montre les différentes étapes du processus de création de la Motion Map à partir d'un champ de vecteurs.

Dans le chapitre précédent nous avons présenté un algorithme de placement général permettant d'obtenir des ensembles de streamlines ayant des densités allant de éparses à denses selon le choix de la distance de séparation. Ces représentations sont sauvegardées sous la forme d'un ensemble de lignes brisées dont on stocke pour chaque sommet, les coordonnées et les informations de couleur et d'épaisseur. La finalité de l'algorithme de la Motion Map est différente. La sortie sera maintenant un tableau de résolution fixée, de même taille que l'image finale et dont chaque cellule aura été traversée par une streamline qui lui aura légué un indice de couleur.

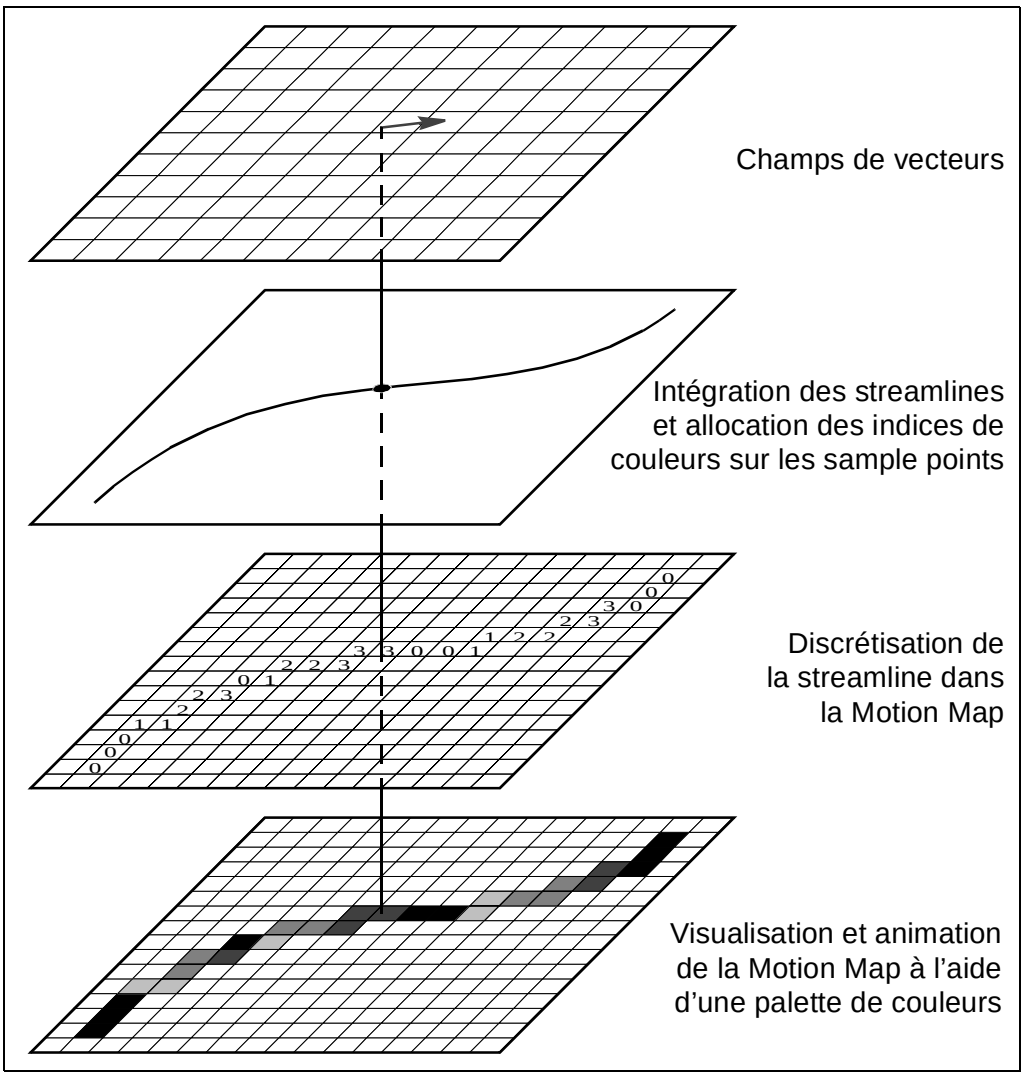

Figure 4.4 Du champ de vecteurs à la Motion Map.

Pour atteindre cet objectif de façon efficace, nous avons adapté l'algorithme de placement général présenté dans le chapitre précédent. L'adaptation a touché à des composants de l'algorithme comme la sélection des seed points et l'arrêt de la croissance des streamlines et nous a amené à considérer des aspects propres au remplissage de la Motion Map. Les sections suivantes décrivent ces modifications.

Auparavant, il convient toutefois de distinguer les différents espaces de travail :

- un espace continu dans lequel est faite l'intégration des streamlines à partir des seed points proposés,
- un espace discret, celui de la Motion Map, dans lequel sont discrétisées les streamlines après avoir été calculées.

3.1 Sélection des seed points

Comme nous désirons couvrir l'ensemble de la Motion Map, toute cellule «libre» est un seed point valide. Une cellule est dite libre lorsqu'aucun indice de couleur ne lui a encore été affecté. Une technique de sélection simple, similaire au placement de streamlines depuis un grille régulière, est de parcourir toutes les cellules de la Motion Map et de proposer la position de leur centre comme seed point quand elles sont libres. On remarque aussi que la Motion Map peut naturellement jouer le rôle du masque comme dans la sélection aléatoire sur masque booléen décrite dans la section 2.3.3 du chapitre 3 :

- les cellules libres de la Motion Map sont les seed points valides (par convention une cellule est libre quand sa valeur est égale à -1),
- la mise à jour du masque est effectuée (quelque soit la méthode de sélection choisie) lors de la discrétisation des streamlines. L'utilisation d'un masque n'est donc plus un coût supplémentaire pour l'algorithme.

Nous avons programmé ces deux méthodes de sélection de seed points. Les temps de calculs sont quasiment identiques, toutefois nous verrons dans la section 3.3 que l'apparition de motifs dûs au parcours régulier de la grille nous fera préférer la technique de sélection aléatoire sur un masque.

3.2 Croissance des streamlines

A partir d'un seed point sélectionné, une streamline est intégrée dans les deux sens sous forme d'une série de sample points. L'intégration n'est pas effectuée dans l'espace discret de la Motion Map mais dans l'espace continu pour minimiser l'accumulation des erreurs qui seraient dues à la discrétisation. L'intégration arrête dans l'un des trois cas suivant :

- lorsque la streamline atteint un bord de la Motion Map,
- ou un point critique,
- ou une cellule non libre de la Motion Map.

Pour la dernière condition d'arrêt, une conversion de la position du nouveau sample point de l'espace continu de l'intégration dans l'espace discret de la Motion Map est nécessaire pour tester l'occupation de la cellule. Remarquons que ce test, qui remplace le calcul de distance de l'algorithme général, est très peu coûteux et permet un gain en temps de calcul de l'ordre d'un facteur 10.

Cependant il arrive fréquemment que l'intégration stoppe prématurément à cause d'une cellule occupée. C'est inévitablement le cas quand une streamline est construite proche d'une autre dont la trajectoire est courbe : le crénelage dû à la discrétisation de la plus ancienne obstrue le passage (voir figure 4.5a). Pour sortir de ces «pièges», nous autorisons une streamline à heurter plusieurs fois de suite des cellules occupées durant sa croissance. L'intégration de la streamline

est arrêtée si n_{heurts} sample points consécutifs se trouvent dans des cellules occupées. Les streamlines sont ainsi plus longues et corrèlent plus de cellules dans la Motion Map (voir figure 4.5b). Nous introduisons une mesure de cette corrélation dans la section 3.3. Avec cette mesure nous avons constaté que la corrélation augmentait avec l'accroissement de n_{heurts}, de même que le temps de calcul (voir figure 4.9). Prendre n_{heurts} = 3 ou 4 donne déjà une corrélation suffisante pour un temps de calcul minimal.

Pour profiter de la totalité de la résolution spatiale qu'offre la Motion Map on fixe l'écartement d_{ecart} entre les sample points de telle sorte que des sample points consécutifs se projettent dans des cellules adjacentes de la Motion Map lors de la discrétisation des streamlines.

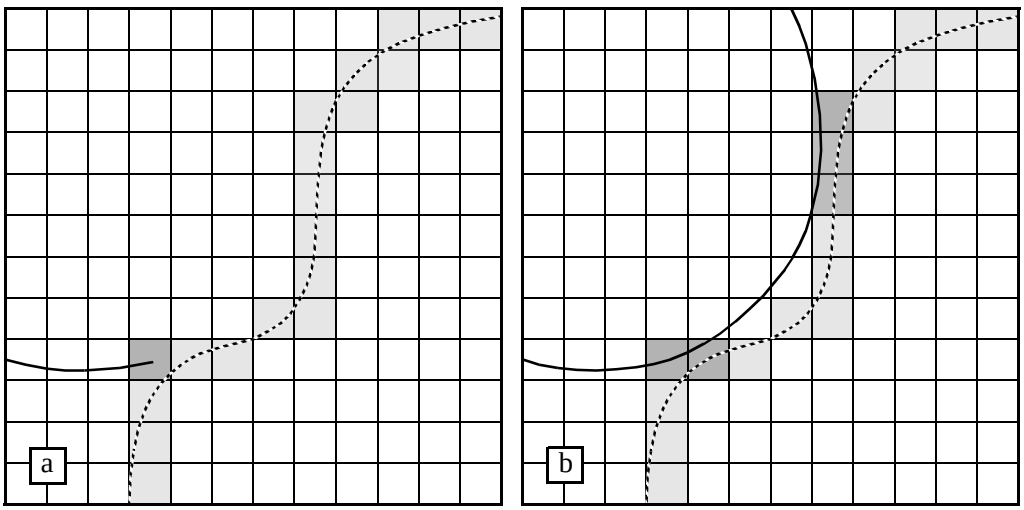

Figure 4.5 (a) Arrêt de l'intégration d'une streamline dû au crénelage d'une streamline précédente. (b) Obtention d'une streamline plus longue en autorisant plusieurs heurts consécutifs de cellules occupées (ici, $n_{heurs} \geq 3$).

3.3 Discrétisation des streamlines dans la Motion Map

Lorsque l'intégration d'une streamline est terminée, on parcours les sample points pour leur allouer un indice de couleur comme nous l'avons décrit dans la section 2.2. Chaque segment de la streamline est alors discrétisé dans la Motion Map. La valeur affectée à chaque cellule traversée par un segment correspond à l'indice porté par le premier sample point du segment (voir figure 4.6a).

Pour évaluer la qualité des représentations obtenues, nous nous sommes doté d'une mesure de la corrélation des cellules de la Motion Map (voir section 3.3.1). Grâce à cette mesure, nous avons significativement amélioré la corrélation en modifiant l'algorithme de discrétisation initial (voir section 3.3.2).

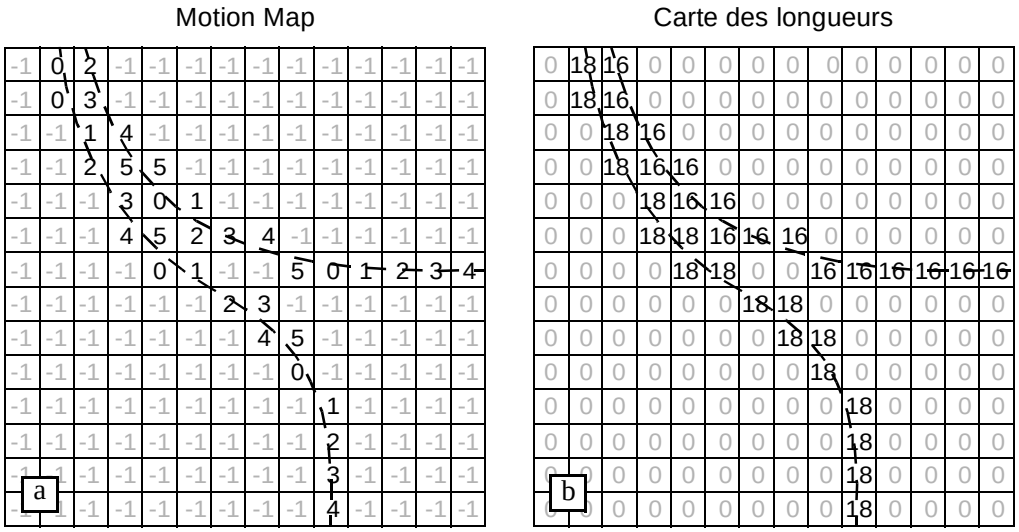

Figure 4.6 **Discrétisation des streamlines dans la Motion Map.** (a) Les indices alloués aux sample points sont affectés aux cellules de la Motion Map traversées par les streamlines. (b) la longueur des streamlines est stockée dans une carte des longueurs.

3.3.1 Mesure de la corrélation des cellules de la Motion Map

Plus le nombre de cellules corrélées entre elles est important et plus les animations seront de qualité. Cette corrélation se fait le long de chaque streamline. Ainsi plus les portions ininterrompues de streamlines sont longues et plus la corrélation est bonne. Pour mesurer cette corrélation, nous avons superposé à la Motion Map un tableau de même résolution dont chaque cellule contient la longueur de la streamline qui a alloué un indice à cette position dans la Motion Map (voir figure 4.6b). Cette carte rassemble donc les longueurs de toutes les streamlines ayant contribué à la construction de la Motion Map.

Une mesure quantitative de la corrélation des cellules de la Motion Map peut être obtenue en faisant la moyenne des longueurs de la carte. Une mesure qualitative est obtenue en visualisant directement la distribution des longueurs dans la carte en leur affectant un niveau de gris. La mesure quantitative n'est cependant pas absolue : elle dépend de la résolution de la Motion Map (plus la résolution est élevée et plus les streamlines ont une chance d'être longues) et de la turbulence du champ de vecteurs (un flux en «spiral» peut avoir des streamlines plus longues qu'un flux laminaire). Elle est toutefois fiable pour mesurer un accroissement de corrélation entre deux méthodes sur une même Motion Map et pour un champ de vecteurs donné.

3.3.2 Adaptation de l'algorithme de discrétisation

Initialement, nous avons utilisé l'algorithme de Bresenham [23] pour tracer chaque segment de streamline dans la Motion Map. Dans cette première version, toutes les cellules se trouvant entre deux sample points reçoivent l'indice de couleur affecté au premier. Rappelons que des recouvrements sont possibles car le nombre de heurts n_{heurts} autorise une streamline à passer par une cellule déjà affectée. L'indice dans chaque cellule est finalement celui que lui a légué la dernière streamline qui l'a traversée.

Pour cet algorithme, la visualisation de la carte des longueurs montre une distribution très chaotique (voir figures 4.7a et 4.7d). Les longues streamlines (streamlines claires) sont fréquemment coupées par des streamlines plus courtes (streamlines sombres), ce qui diminue la corrélation le long des longues streamlines.

Une première amélioration de la corrélation a consisté à modifier l'algorithme de Bresenham en partant de l'intuition suivante : les premières streamlines ont plus de chance d'être longue puisqu'elles ont plus de place pour croître. Par conséquent, le tracé d'un segment ne pourra affecter que les cellules libres de la Motion Map. Cette condition simple a permis une augmentation très significative de la longueur moyenne, ainsi que de la distribution des longueurs, comme en atteste les figures 4.7b et 4.7e.

Finalement, nous avons optimisé la discrétisation des streamlines en nous servant de la carte des longueurs dans l'algorithme de tracé des segments. Maintenant, une cellule de la Motion Map ne peut être affectée que si la longueur de la streamline à discrétiser est supérieure à la longueur courante en cette position. L'amélioration n'est pas très visible à l'oeil nu sur la distribution des longueurs mais l'augmentation de la moyenne des longueurs atteste de l'augmentation de la corrélation (voir figures 4.7c et 4.7f).

Les résultats de la figure 4.7 ont été obtenus pour les deux méthodes de sélection de seed points présentées dans la section 3.1. On remarque une légère supériorité des moyennes de longueurs pour la méthode de sélection dite «grille». On peut toutefois noter les coupures indésirables des streamlines, signalées par les cercles. On remarque que ces coupures n'apparaissent pas avec la méthode de sélection dite «masque». Une illustration encore plus probante de la différence de qualité dans les représentations obtenues est donné avec la figure 4.8.

Enfin, une dernière remarque concerne le choix de la valeur de n_{heurts} abordé dans la section précédente. La figure 4.9 montre que la corrélation (longueur moyenne dans la carte) et le temps de calcul croissent avec le nombre de heurts. Un compromis est donc à faire entre temps de calcul et qualité des représentations. Remarquons cependant que les représentations obtenues avec n_{heurts} = 3 ou 4 sont déjà de très bonne qualité.

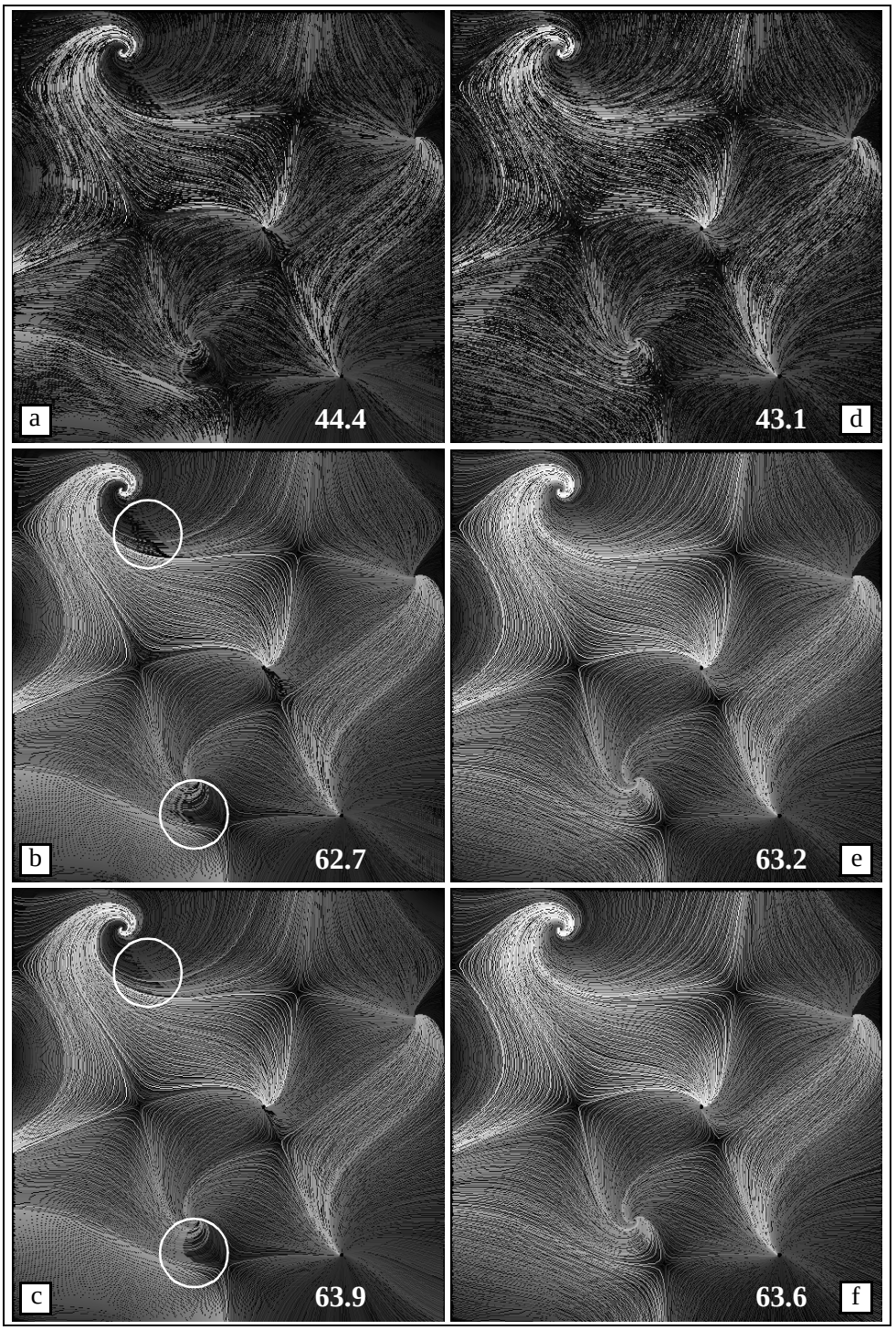

Figure 4.7 **Carte des longueurs** pour chaque cellules de la Motion Map obtenues avec (a,d) la version originale de l'algorithme de Bresenham, (b,e) en n'affectant que les cellules libres et (c,f) en laissant la priorité aux streamlines les plus longues. Sélection de seed points sur (droite) «grille» (gauche) «masque». Les nombres donnent la moyenne des longueurs pour chaque carte.

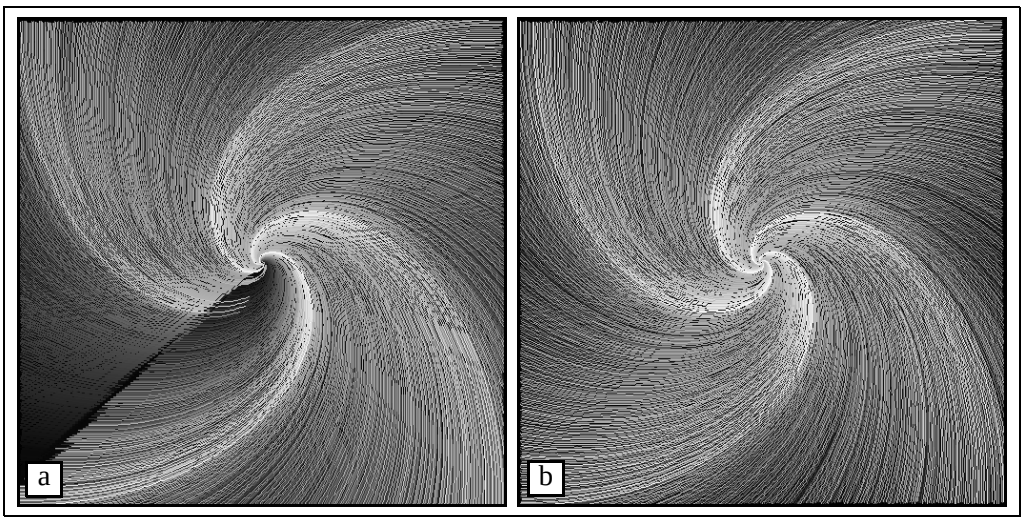

Figure 4.8 (a) Apparition de coupures indésirables dans les streamlines avec la sélection de seed points sur une grille. (b) Ces artefacts n'apparaissent pas avec la sélection des seed points aléatoire sur un masque.

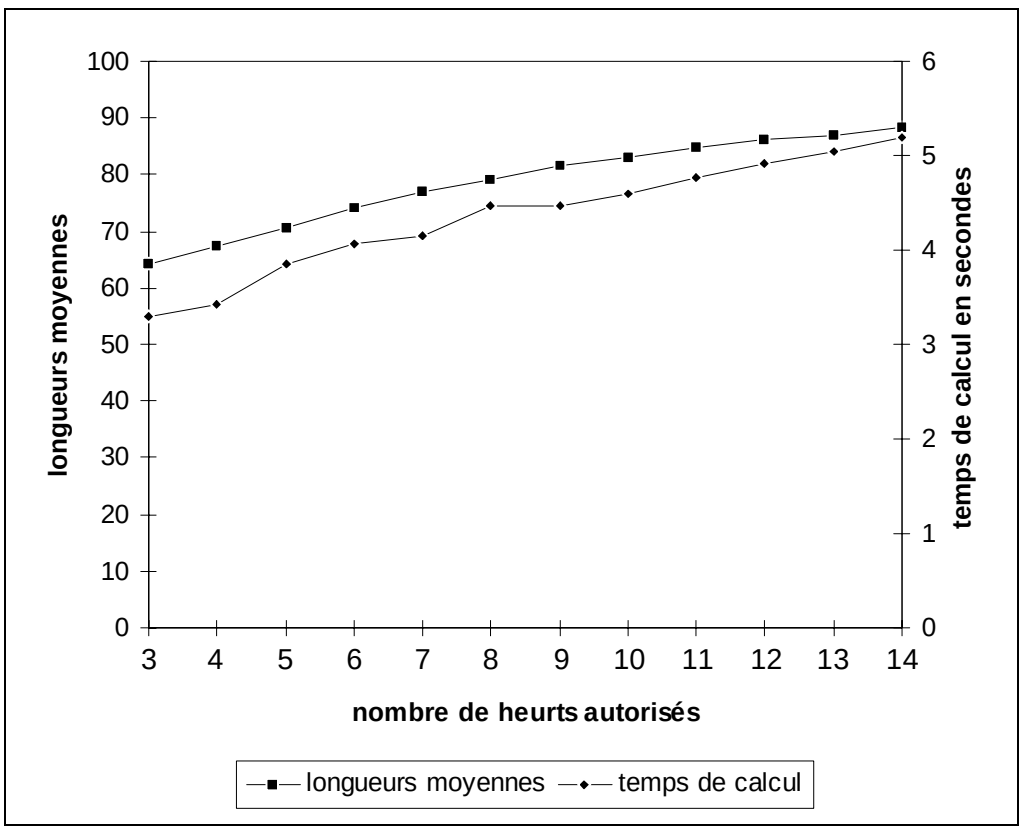

Figure 4.9 Influence du nombre de heurts autorisé n_{heurts} sur la longueur moyenne dans la carte des longueurs et le temps de calcul.

4 Visualisation de l'animation

Une fois la Motion Map remplie, on visualise le flux en animant son contenu. Les deux sections suivantes présentent deux méthodes pour visualiser une animation cyclique dérivée de la structure de la Motion Map. La première est l'utilisation de la technique simple et efficace de l'animation de la palette des couleurs, la seconde consiste à dériver un ensemble cyclique de textures.

Notons qu'il est possible d'avoir une bonne idée de la structure du flux en observant une image statique obtenue en remplaçant les indice de la Motion Map par les couleurs d'une table des couleurs. Les représentations obtenues ressemblent alors à celles produites par LIC ou SpotNoise.

4.1 Animation de la palette des couleurs

Nous avons proposé un algorithme d'allocation des indices de couleurs dans la Motion Map de sorte que l'application de la technique d'animation par la palette des couleurs soit immédiate. Il suffit en effet de fournir une palette de N+1 couleurs dont N couleurs représentent le motif qui sera déplacé sur les streamlines, on les appelle les *couleurs animées* de la palette, la couleur restante est la couleur fixe utilisée pour les pixels par lesquels n'est passée aucune streamline (à ces endroits, l'indice dans la Motion Map vaut -1). L'image est alors affichée avec les couleurs de la palette et son apparence est continuellement remise à jour en décalant circulairement les N couleurs animées de la palette.

Comme décrit dans [21], en jouant sur les couleurs allouées, il est possible d'étendre le principe de la Motion Map de façon à visualiser, en plus de l'orientation du flux, une valeur scalaire indépendante telle qu'une pression, une température ou un niveau d'énergie par exemple. Le mouvement est codé par des variations d'intensité alors que la valeur scalaire est codée par un ensemble de teintes. Ces informations sont stockées dans une palette de couleurs qui est divisée en plusieurs segments de même longueur. On associe à chaque segment une teinte que l'on dérive selon plusieurs intensités pour remplir les cellules du segment de palette. Le nombre de segments détermine le nombre de teintes distinctes disponibles pour la valeur scalaire alors que la longueur des segments donne le nombre d'intensités codant le mouvement. Avec une palette de couleur standard de 256 couleurs on peut réserver 16 intensités pour le mouvement et 16 couleurs distinctes pour la valeur scalaire (ou 6 intensités et 42 teintes dans une palette de 252 couleurs par exemple). Pour construire la Motion Map, on fixe N au nombre d'intensités nécessaires pour coder le mouvement et à partir des indices donnés par la formule (4.3) on ajoute un décalage dû à la valeur scalaire de façon à référencer le segment de

teinte correspondant. Une méthode générale pour calculer une telle Motion Map à partir d'un champ de vecteurs et d'une image représentant le contexte de l'animation (fond de carte par exemple) a été proposé par Chédot et Lefer [10].

La Motion Map et sa palette de couleurs peuvent être stockées dans des formats de fichiers d'images classiques tel que le format GIF. L'image statique du flux peut alors être visualisée par n'importe quel programme d'affichage d'images. Moyennant que l'on dispose d'un programme capable d'animer la palette des couleurs, la Motion Map est le moyen le plus efficace pour stocker une animation de flux stationnaire, puisque la taille de l'animation complète se réduit à la taille d'une seule image.

4.2 Ensemble cyclique de textures

La Motion Map peut aussi être exploitée pour produire un ensemble cyclique de textures qui peuvent être assemblées dans un format d'animation classique (mpeg, avi, ...) ou être plaquées sur des objets 3D complexes dans un système de visualisation. Un ensemble de textures a une utilisation plus large que l'animation de la table des couleurs qui impose de travailler en mode fausses-couleurs avec un nombre de couleurs limité. Les textures que nous dérivons de la Motion Map sont en mode vraies couleurs (RGB) et sont obtenues en multipliant des intensités associées aux indices de la Motion Map avec une carte de couleur de même résolution, qui peut être utilisée par exemple pour coder une valeur scalaire indépendante.

Soit M une Motion Map avec $M(x,y) = i, \quad i \in \{-1, \ldots, N-1\}$, et $Intens(i) : \{0, \ldots, N-1\} \rightarrow [0;1]$ une fonction discrète qui associe une intensité à chaque indice positif de la Motion Map. Cette fonction donne la forme des motifs qui se déplacent sur les streamlines. Soit C_{rgb} une carte des couleurs où $C_{rgb}(x,y)$ est un triplet codant la couleur en mode RGB représentant la valeur scalaire indépendante à l'emplacement (x,y). Une texture colorée T_{rgb} est obtenue en multipliant les couleurs par les intensités :

$$T_{rgb}(x,y) = \begin{cases} Intens(M(x,y)) \times C_{rgb}(x,y) & \forall M(x,y) \geq 0 \\ D_{rgb}(x,y) & \forall M(x,y) < 0 \end{cases} \tag{4.9}$$

où $D_{rgb}(x,y)$ est une fonction qui donne une couleur RGB aux endroits où aucune streamline n'a été intégrée (indice la Motion Map égal à -1). Par défaut on prend $D_{rgb}(x,y) = C_{rgb}(x,y)$.

L'ensemble des N textures $T^t_{rgb}, t \in \{0, \ldots, N-1\}$ est obtenu en effectuant pour chaque texture un décalage des indices contenus dans la Motion Map :

$$T^t_{rgb}(x,y) = \begin{cases} Intens((M(x,y) + t) \bmod N) \times C_{rgb}(x,y) & \forall M(x,y) \geq 0 \\ D_{rgb}(x,y) & \forall M(x,y) < 0 \end{cases} \tag{4.10}$$

La plus simple configuration pour percevoir le mouvement du flux contenu dans une Motion Map M(x,y) étant de choisir les paramètres suivants pour dériver les textures d'après l'équation (4.10) :

$$\begin{cases} Intens(i) = i/(N-1) \\ C_{rgb}(x,y) = D_{rgb}(x,y) = (255, 255, 255) \qquad \forall x \forall y \end{cases} \tag{4.11}$$

5 Résultats et discussions

La méthode d'allocation de couleurs que nous avons exposé dans ce chapitre peut parfaitement être utilisée pour n'importe quels ensembles de streamlines, qu'ils soient éparses ou denses, et en particulier avec les ensembles obtenus avec notre algorithme de placement général. Nous avons cependant adapté cet algorithme de placement afin d'accélérer le calcul des représentations denses. En remplaçant le calcul de distance entre un sample point et les streamlines existantes par un test d'occupation des cellules de la Motion Map (voir section 3.2), nous avons pu accélérer les calculs de l'ordre d'un facteur 10.

Nous avons développé les deux méthodes de visualisation proposées dans les sections 4.1 et 4.2. Les deux paragraphes suivants évoquent l'inflence du choix de la palette des couleurs sur les animations produites et l'utilisation de textures semi-transparentes dans le cas du plaquage de textures en 3D. Enfin, nous proposons une brève comparaison qualitative de notre approche avec les autres techniques d'animation de champs de vecteurs proposées dans la littérature.

Choix de la palette des couleurs

Le choix de la palette des couleurs détermine l'allure des motifs qui se déplacent sur les streamlines. Ce choix est déterminant pour l'apparence finale de l'image (et donc de l'animation) obtenue. La figure 4.10 montre 3 représentations calculées à partir de la même Motion Map mais avec des palettes différentes. Les palettes contiennent 32 couleurs, les deux premières sont fixes, les autres sont animées. Quand la Motion Map est animée avec la palette c, des particules semblent se déplacer le long du flux. Cependant cette palette n'est guère intéressante pour la visualisation du flux sous forme d'une image fixe car elle ne permet pas d'en interpréter la structure. Notons que dans le cas de la visualisation de l'animation par la technique d'animation de la palette des couleurs, la palette peut être changée pendant l'animation, permettant ainsi de choisir celle qui correspond le mieux au résultat souhaité.

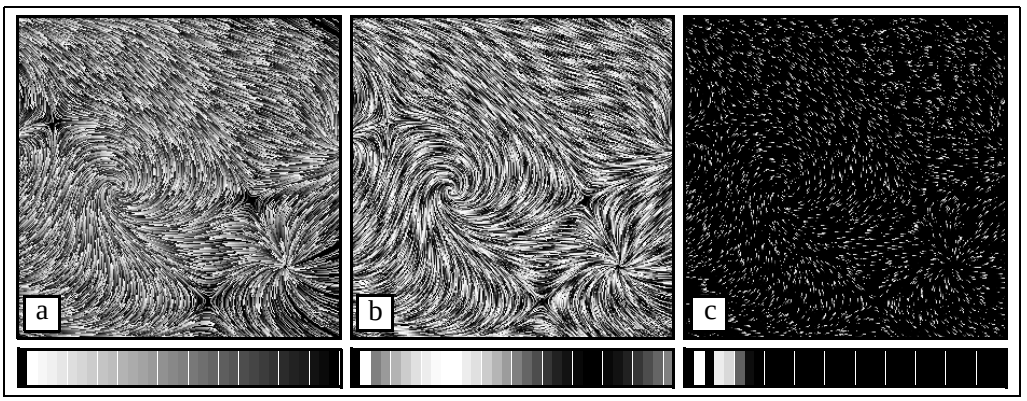

Figure 4.10 Influence de la palette des couleurs sur la représentation d'un champ de vecteurs. Les images ont été obtenues avec 3 types de palette différentes. (a) discontinue de type «dent de scie», (b) continue de type «sinusoïdale», (c) éparse de type «particule».

Textures semi-transparentes

Nous avons développé un logiciel en OpenGL permettant de visualiser une Motion Map sous forme de textures semi-transparentes. La technique de l'ensemble cyclique de textures décrit dans la section 4.2 est utilisé pour extraire les *N* textures de la Motion Map auxquelles on ajoute une information de transparence proportionnelle à l'intensité de leur pixels. Il suffit de plaquer successivement les textures sur des primitives graphiques pour voir le flux évoluer à leur surface. L'animation en temps réel est atteinte grâce aux accélérateurs 3D des cartes graphiques.

L'avantage de cette technique est qu'il est possible d'inclure une animation de flux à l'intérieur de scènes 3D. Nous avons appliqué cette technique pour visualiser un champ de vecteurs stationnaires des vents autour du globe terrestre. Les textures sont plaquées sur une sphère de rayon légèrement supérieur à celui de la terre. La transparence permet d'apercevoir les continents à travers les courants atmosphériques pendant que la terre tourne autour de son axe (voir figure 4.11).

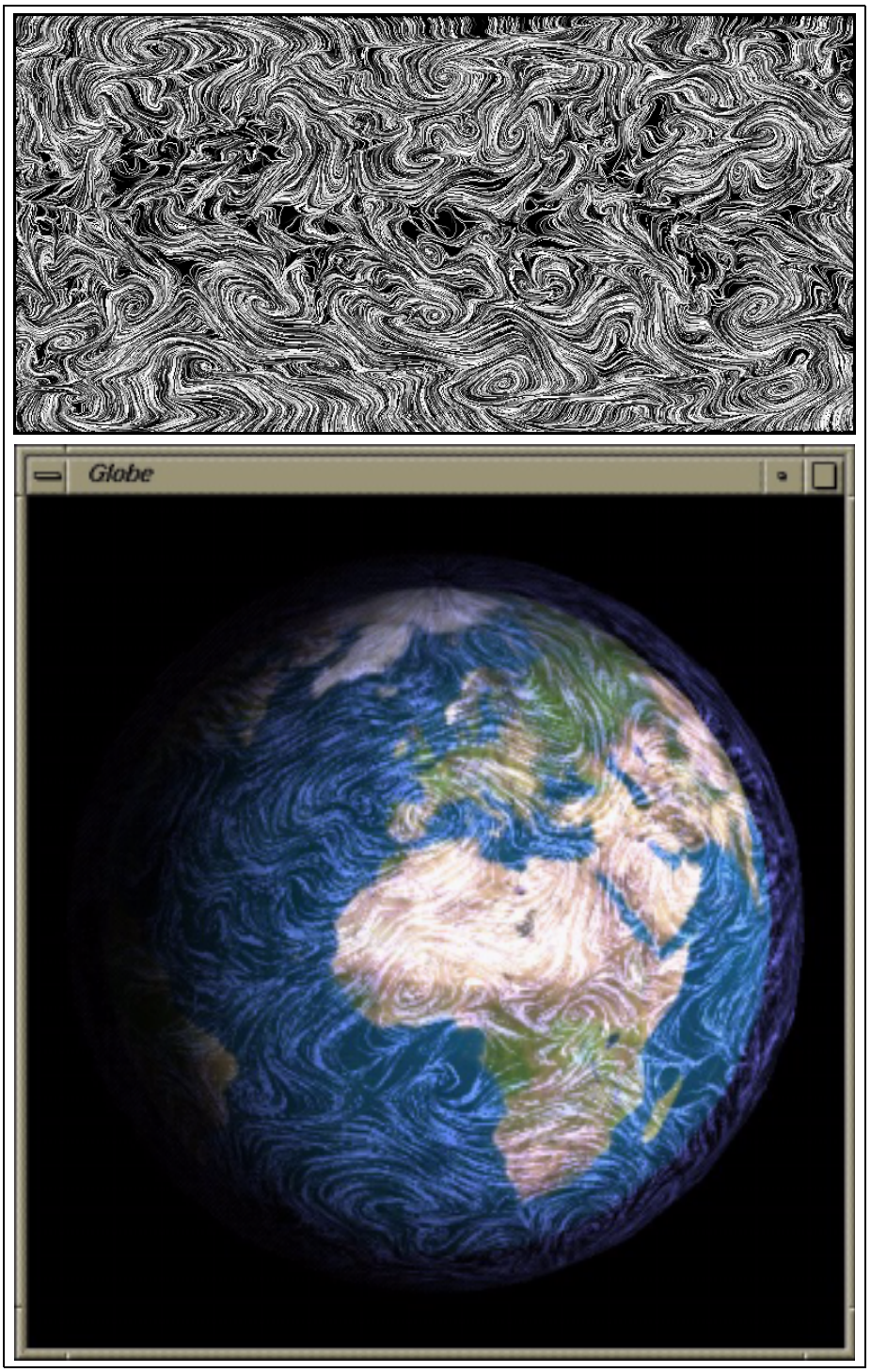

Figure 4.11 **Prédiction des vents sur le globe terrestre**. (haut) Motion Map calculée d'après un champ de vecteurs des vents au dessus de la terre, (bas) Plaquage d'un ensemble de textures semi-transparentes dérivées de la Motion Map autour du globe terrestre et animation. Données originales fournies par le Centre Météorologique Canadien.

5.1 Comparaison avec *LIC* et *SpotNoise*

Pour terminer ce chapitre, il convient de comparer notre approche avec les deux techniques les plus reconnues dans la littérature pour la production de représentations denses de champs de vecteurs 2D stationnaires. LIC et SpotNoise, initialement présentées pour la création d'images statiques ont toutes les deux été étendues pour produire des animations (se référer au chapitre 2 pour plus de détails sur ces techniques).

Il a été montré que LIC et SpotNoise sont semblables sous beaucoup d'aspects [15]. Nous retiendrons principalement que les mécanismes de corrélation spatiale utilisés ont un support limité (une vingtaine de pixels) correspondant à la longueur du noyau de convolution pour LIC et à la longueur du spot pour SpotNoise. En comparaison, nous avons vu que notre approche tendait a maximiser la corrélation des pixels en favorisant les supports (streamlines) les plus longs (une soixantaine de pixels pour les figures 4.7c et 4.7f). Cette différence est particulièrement appréciable pour la création d'animations, car dans notre cas, plus la longueur des supports est grande et plus les corrélations spatiales et temporelles seront élevées.

5.1.1 Représentations statiques

Bien qu'elle n'ait pas été conçue à cet effet, l'obtention d'une représentation statique a partir de la Motion Map ne nécessite que l'adjonction d'une table de couleurs. Les images des figures 4.3 a et b, 4.10 et 4.11a ont été créées de cette façon. En comparant la qualité visuelle par rapport aux représentations denses obtenues par LIC et SpotNoise on remarque que nos images sont moins attrayantes car beaucoup plus aliassées. De plus le temps de calcul pour la Motion Map semble légèrement supérieur à ceux nécessaires pour la génération d'une image LIC ou SpotNoise.

Pour la création de représentations statiques plus attrayantes avec le principe de la Motion Map, une amélioration consisterait à allouer directement des couleurs à ses cellules (et non des indices de couleurs) et à tracer des streamlines anti-aliasées de façon à moyenner les teintes des pixels adjacents. Notons que par ce principe, la structure de la Motion Map ne pourrait plus contenir une animation.

5.1.2 Animations

Les deux techniques utilisent, dans leur versions les plus abouties [24] et [14], (animations cycliques avec représentation des vitesses variables) un mécanisme de fondu (*blending*) pour garder une corrélation temporelle acceptable d'une image à l'autre de l'animation. Ce fondu entraîne une perte de contraste temporel (les motifs qui se déplacent ont des frontières floues) et par conséquent une diminution de la résolution spatiale. En comparaison avec la Motion Map, cette dernière méthode produit des animations beaucoup plus contrastées et de résolution spatiale plus élevée. Enfin, au niveau du temps de

calcul et du stockage, notre méthode et beaucoup plus avantageuse puisqu'une seule «image» contient toute l'information nécessaire à une animation de *N* images. Pour LIC et SpotNoise, *N* images différentes sont requises.

6 Conclusion

Dans ce chapitre nous avons présenté une méthode permettant de produire des animations pour visualiser des champs de vecteurs bidimensionnels stationnaires. La principale inovation de cette approche réside dans la création de la structure de données, la Motion Map, qui a la taille d'une image bitmap et qui permet de stocker toutes les informations permettant d'animer le flux.

La Motion Map est remplie en discrétisant un ensemble dense de streamlines sur lesquelles des indices de couleurs ont été alloués. C'est le déplacement des motifs de couleurs sur les streamlines qui donne l'effet de mouvement du flux. Nous avons proposé deux modes d'animation de la Motion Map. Le premier est l'utilisation de l'animation de la palette des couleurs qui permet une animation en temps réel sur des ordinateurs bas de gamme. La seconde utilise les capacités graphiques de plaquage de textures des ordinateurs pour projeter de façon cyclique un ensemble de textures dérivé de la Motion Map.

Bien que basé sur notre algorithme de placement général de streamlines présenté dans le chapitre 3, l'adaptation au cas dense nous a permis de réduire les temps de calcul d'un facteur 10. De plus, en nous dotant d'une mesure de la corrélation spatio-temporelle des représentations générées, nous avons pu concevoir un algorithme optimisant la corrélation des animations produites.

Par comparaison visuelle avec les animations produites par les autres méthodes existantes, telles que LIC ou SpotNoise, nous pouvons affirmer que la Motion Map est à ce jour, la méthode produisant les meilleures animations de champs de vecteurs 2D stationnaires.

Une partie de ce travail sur la Motion Map à fait l'objet d'une publication à la conférence IEEE Visualization'97 à Phoenix (Arizona) [31].

CHAPITRE

5

Visualisation de champs de vecteurs non stationnaires

1 Introduction

Ce chapitre présente l'extension de notre travail sur le placement de streamlines à la visualisation de champs de vecteurs non stationnaires. Jusqu'à présent notre travail à porté sur la visualisation de flux stationnaires qui par nature ont une structure invariante au cours du temps. Contrairement à ces flux qui ne nécessitent qu'un seul champ de vecteurs pour les représenter, les flux non stationnaires sont définis par une série de champs de vecteurs représentant un échantillonnage temporel de la structure du flux. Les objectifs de la visualisation de tels champs de vecteurs sont les mêmes que dans le cas stationnaire, à savoir pouvoir interpréter la direction, l'orientation, la vitesse et la position des points critiques. La différence principale avec les flux stationnaires étant que ces caractéristiques évoluent au cours du temps. La visualisation d'un flux non stationnaire nécessite alors la production d'animations afin de suivre l'évolution de ces caractéristiques.

Les travaux sur les champs non stationnaires s'appuient pour la plupart sur des streaklines ou des pathlines, courbes dépendant du temps, pour élaborer leur technique de visualisation. C'est particulièrement le cas des extensions de LIC [18][49] où les streamlines qui portaient les noyaux de convolution sont remplacées par des pathlines (voir section 6.4 du chapitre 2). Nous proposons une alternative à ces travaux en utilisant des streamlines, courbes ne dépendant pas du temps, pour visualiser les flux non stationnaires. L'idée est de représenter ces flux en visualisant la succession de leurs états instantanés décrits par des ensembles de streamlines calculés à chaque pas de temps. La dimension temporelle intervient dans notre approche dans la corrélation de ces états instantanés successifs, c'est à dire dans la corrélation des ensembles de streamlines intégrées dans les champs de vecteurs consécutifs. Le processus de

corrélation des streamlines nécessite de trouver d'un pas de temps à l'autre un ensemble de streamlines qui ressemblent le plus, en forme et en position, avec celles de l'ensemble du pas de temps précédent de manière à ce que la déformation de la structure du flux soit progressive et continue. Pour garder une bonne interprétation de la structure du flux au cours du temps, nous avons imposé aux streamlines de respecter un critère de densité fixé.

La technique que nous proposons permet aussi de visualiser l'orientation du flux au cours de l'animation par le déplacement de textures sur les streamlines. Là encore, un processus de corrélation est nécessaire pour que le mouvement des textures au cours du temps soit le plus fluide possible. Contrairement à la Motion Map dédiée aux flux stationnaires, le mécanisme de corrélation des textures doit ici s'adapter à des streamlines qui changent de position, de forme et de longueur.

Les principales caractéristiques de la méthode que nous proposons dans ce chapitre sont :

- placement optimisé des streamlines dans chaque image, basé sur l'algorithme de placement présenté dans le chapitre 3.
- codage du mouvement le long de chaque streamline en plaquant des textures se déplaçant au cours du temps afin de visualiser l'orientation du flux. Nous avons développé une autre technique que celle décrite pour la Motion Map (chapitre 4) de façon à s'adapter à l'évolution de la structure des flux non stationnaires.
- corrélation des images successives de l'animation en utilisant un algorithme de corrélation des streamlines. Aussi bien la forme que les textures des streamlines sont corrélées.

Ce chapitre est organisé de la façon suivante. La section 2 décrit la manière dont les streamlines sont corrélées d'une image à l'autre afin d'obtenir une évolution fluide de la structure du flux au cours du temps. La section 3 montre comment l'orientation du flux est visualisée par le déplacement de textures corrélées sur les streamlines. Des résultats sont présentés dans la section 4 avant de conclure par la section 5.

2 Corrélation des streamlines au cours du temps

La corrélation des streamlines dans les images successives de l'animation est une opération nécessaire pour obtenir une animation fluide de la structure du flux. Sans cette corrélation, c'est à dire en calculant pour chaque pas de temps un ensemble de streamlines sans tenir compte des streamlines précédentes, il n'est pas difficile d'imaginer que les streamlines se succéderaient aléatoirement dans l'animation produisant un effet de clignotement rendant difficile l'interprétation du comportement du flux.

D'après des observations faites sur des animations de test, nous avons identifié trois principaux évènements visuels ayant tendance à dégrader la qualité de l'animation en terme de capacité pour un observateur à interpréter le comportement du flux. Ces évènements visuels sont classés ci-dessous par impact croissant sur la fluidité de l'animation :

- déformation notable entre deux streamlines *correspondantes*, c'est à dire entre streamlines à la même place dans deux images consécutives,
- disparition de streamlines,
- apparition de nouvelles streamlines (non corrélées).

Pour obtenir des animations fluides de la structure du flux, nous nous sommes attachés à minimiser l'amplitude de ces différences visuelles entre les ensembles successifs de streamlines.

2.1 Aperçu de la méthode

A un instant donné, le nombre de streamlines qu'il est possible de construire à partir du champ de vecteurs est potentiellement infini. L'objectif de l'algorithme de placement est de choisir un sous-ensemble de streamlines corrélées avec les streamlines précédentes tout en respectant le critère de densité propre à la notion de placement. Pour trouver le meilleur ensemble de streamlines corrélées à chaque pas de temps *t*, notre algorithme utilise les streamlines calculées à l'instant *t-1*. La première image de l'animation est constituée des streamlines placées par l'algorithme de placement pour les champs de vecteurs stationnaires qui est décrit dans le chapitre 3 de cette thèse (section 2.4). Pour les images suivantes notre algorithme procède en deux phases.

Dans une première phase, on sélectionne parmi des streamlines calculées au pas de temps *t* toutes celles qui correspondent le mieux aux streamlines du pas de temps *t-1* au regard d'un critère qui mesure la différence entre deux streamlines en terme de position et de forme. Par la suite nous appellerons *streamline de référence* une streamline placée à l'instant *t-1* et *streamline correspondante* une

streamline qui a été sélectionnée à l'instant *t* comme ressemblant à sa streamline de référence.

La seconde phase de l'algorithme consiste à placer de nouvelles streamlines de façon à atteindre une représentation uniforme ayant la densité souhaitée. Ces nouvelles streamlines sont calculées au pas de temps *t*. La figure 5.1 montre les différentes étapes de notre algorithme. Les deux sections suivantes décrivent ces étapes en détails et la section 2.4 propose des moyens supplémentaires permettant d'améliorer la qualité des animations obtenues.

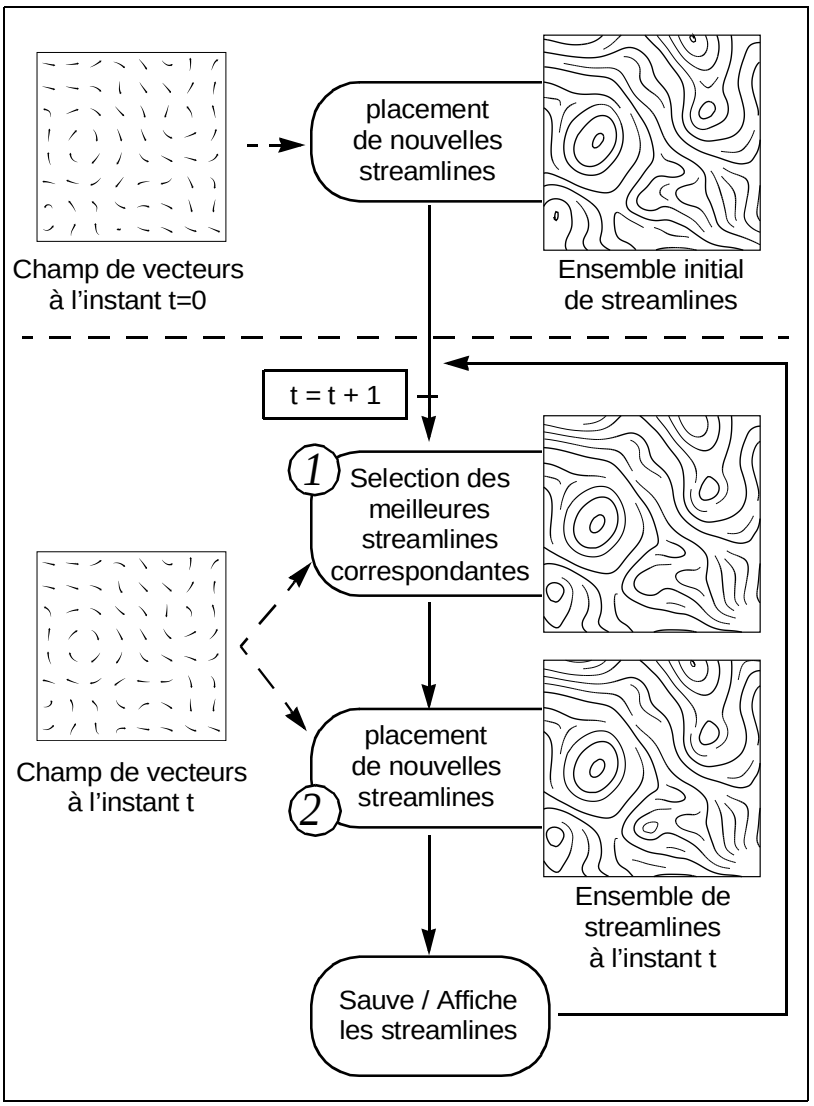

Figure 5.1 Etapes de l'algorithme de placement de streamlines pour les champs non stationnaires.

Une manière intuitive de comprendre la fonctionnement de notre algorithme est de visualiser les différents ensembles de streamlines générés pour passer d'une image du flux à l'instant *t*-1 à l'instant *t*. La figure 5.2a montre un ensemble de streamlines calculé à l'instant *t*-1, on les appelle les streamlines de *référence*. Des streamlines *candidates* sont alors intégrées dans le champ de vecteurs à l'instant *t* (voir figure 5.2b). Parmi ces streamlines, on choisit les streamlines *correspondantes*, «ressemblant» le plus aux streamlines de référence (figure 5.2c). Enfin on ajoute un certain nombre de *nouvelles* streamlines afin que la densité de streamlines dans l'image finale reste constante au cours de l'animation (voir figure 5.2d).

Les sections suivantes expliquent en détail les transitions entre ces ensembles de streamlines intermédiaires.

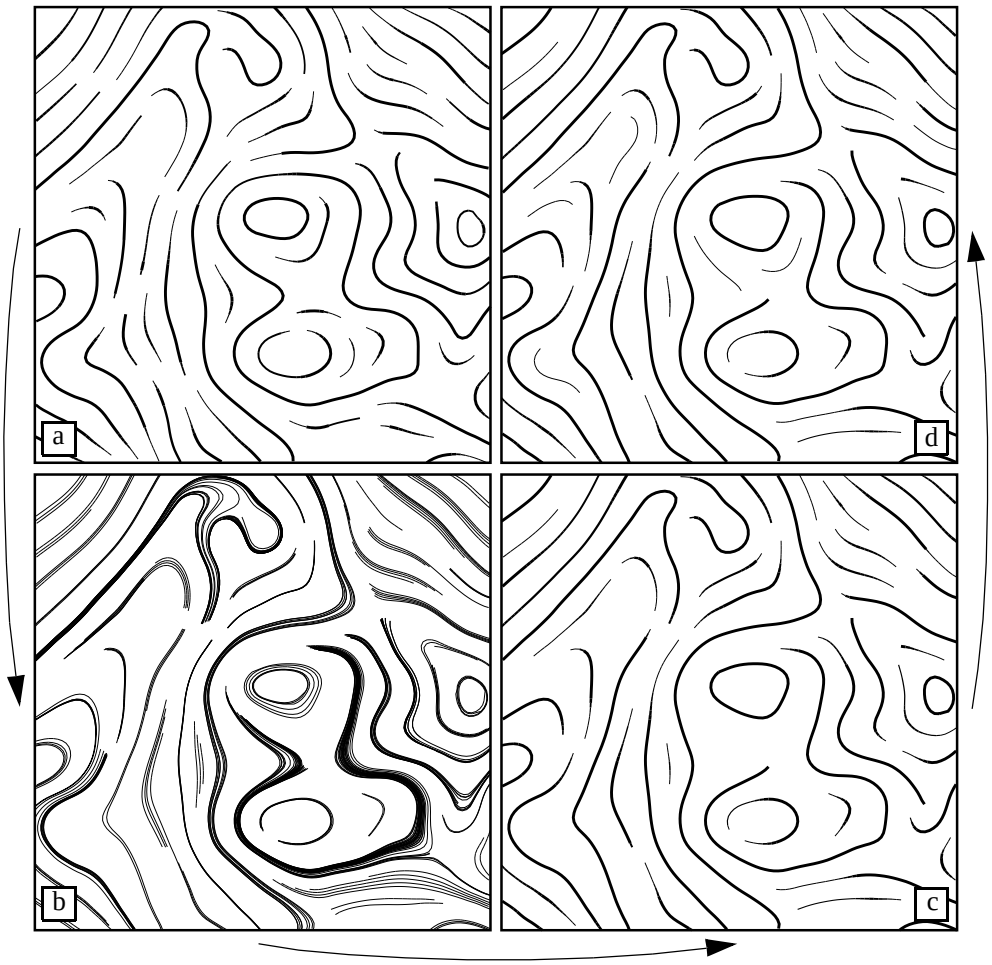

Figure 5.2 Différents ensembles de streamlines nécessaires pour passer d'une image à l'autre de l'animation. (a) streamlines à l'instant *t*-1, (b) streamlines candidates pour l'instant *t*, (c) streamlines correspondantes et (d) ajout de nouvelles streamlines dans l'ensemble final à l'instant *t*.

2.2 Sélection des meilleures streamlines correspondantes

Cette section décrit le passage entre les ensembles de streamlines illustrés par les figures 5.2 a, b et c. Pour chaque streamline placée à l'instant *t*-1, on cherche la streamline correspondante à l'instant *t*. Pour cela, à partir de chacune de ces streamlines de référence, des streamlines candidates sont intégrées au pas de temps *t* à partir de la position de leurs sample points. Ces streamlines candidates sont intégrées parmi les streamlines correspondantes déjà sélectionnées tant que leur distance de séparation ne descend pas au dessous du seuil souhaité. Seuls les streamlines suffisamment longues sont retenues. Il est important de noter qu'à ce stade, toute streamline candidate est valide au regard du critère de densité. Le choix d'initier les streamlines candidates à partir des sample points d'une streamline de référence permet de générer des streamlines dans son entourage et ainsi de favoriser l'apparition de streamlines ressemblantes. Il ne reste plus qu'à sélectionner parmi ces streamlines candidates celle qui obtient le plus haut score au regard du critère de corrélation vis-a-vis de la streamline de référence.

Critère de corrélation entre streamlines

Pour élaborer notre critère de corrélation entre une streamline de référence et une streamline candidate, nous avons considéré leurs *portions communes*. Une streamline candidate étant initiée à partir d'un sample point de la streamline de référence, nous appelons «portions communes» de ces deux streamlines, tous les sample points pouvant être arrangés par paire en partant de ce sample point commun (voir figure 5.3). Nous avons défini le critère de corrélation comme la distance moyenne entre ces paires de sample points. Plus la distance moyenne entre ces portions communes est faible, plus le score de corrélation entre les deux streamlines est élevé.

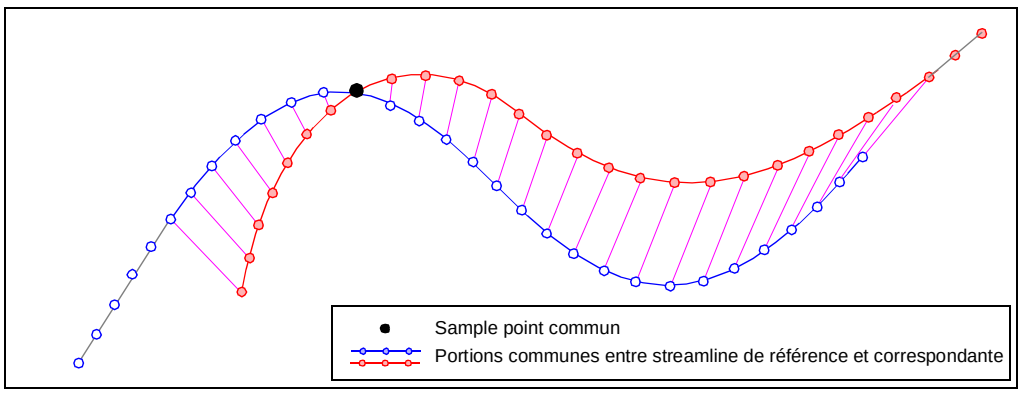

Figure 5.3 Evaluation de la corrélation de deux streamlines sur leur portions communes.

Ce critère nous permet de comparer les deux streamlines sur toute la longueur de leur portion commune. Ainsi une streamline candidate dont la portion commune aura tendance à s'éloigner de la streamline de référence obtiendra un score moins élevé qu'une streamline candidate qui reste plus prés de la streamline de référence. Une fois que la streamline correspondante a été sélectionnée comme meilleure streamline candidate, elle est insérée dans l'ensemble des streamlines placées pour ce pas de temps.

En fait, une streamline de référence peut donner naissance à plusieurs streamlines correspondantes. Ce cas se produit quand une streamline plus courte que la streamline de référence a obtenu la meilleure note sur leur portion commune vis-à-vis des autres streamlines candidates. Cette streamline correspondante est insérée dans l'ensemble des streamlines placées. Il est alors possible que le reste de la streamline de référence puisse donner naissance à une (ou plusieurs) autre(s) streamline(s) correspondante(s) respectant le critère de densité. Pour les générer toutes, une streamline de référence sert à la génération de streamlines candidates tant qu'elle donne naissance à une streamline correspondante.

L'ordre dans lequel sont parcourues les streamlines de référence pour générer les streamlines correspondantes est celui dans lequel les streamlines de références ont été construites au pas de temps précédent. Ainsi si des streamlines ont déjà été placées au moment de la construction d'une streamline S à l'instant t, à l'instant t+1 les streamlines correspondant aux premières streamlines de référence seront placées avant la streamline correspondant à S. La conservation de cet ordre de parcours permet de construire les streamlines candidates et à fortiori les streamlines correspondantes, sous des contraintes similaires d'un pas de temps à l'autre. Cette caractéristique permet à notre algorithme de proposer pour chaque streamline de référence des streamlines candidates assez proche par construction. Le taux de corrélation des streamlines correspondantes sélectionnées n'en est que plus élevé. La figure 5.4 montre les streamlines correspondantes calculées pour un ensemble de streamlines de référence donné.

Il est important de noter que le critère de choix des streamlines correspondantes est relatif et que notre algorithme sélectionne les «meilleures» streamlines parmi des ensembles de streamlines candidates au regard de ce critère. Cela ne garantit pas une ressemblance forte avec les streamlines de référence si la structure du flux varie de façon importante entre deux pas de temps successifs (en cas d'échantillonnage temporel insuffisant). Notre méthode assure alors la meilleure correspondance possible au regard de notre critère de corrélation.

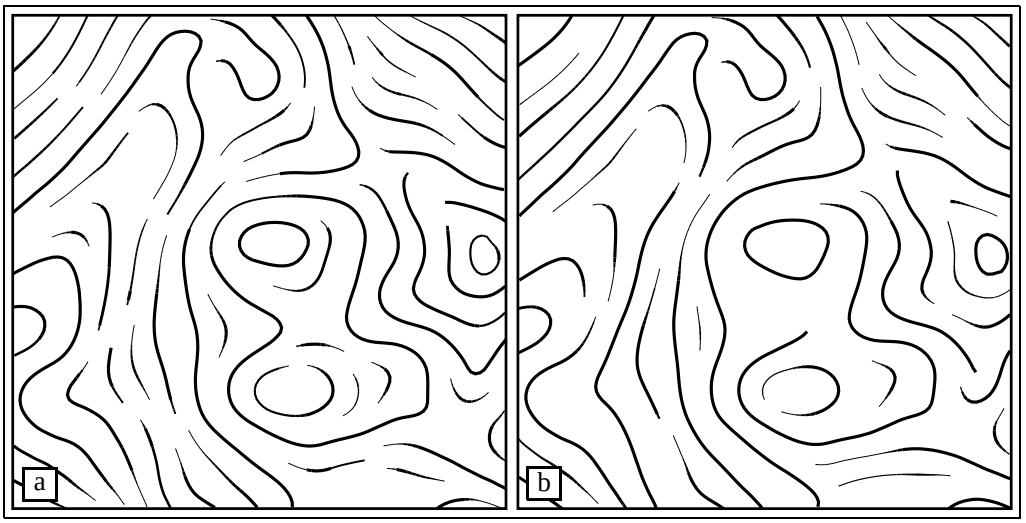

Figure 5.4 (a) Streamlines de références et (b) streamlines correspondantes.

L'algorithme de sélection des streamlines correspondantes est exposé ci-dessous. Les ressources nécessaires à son fonctionnement sont le champ de vecteurs courant et les streamlines calculées au pas de temps précédent.

```
Fonction SélectionnerStreamlinesCorrespondantes(V_t, L_ref, d_sep)
    entrée :  Champ de vecteurs V_t à l'instant t
              Liste de streamlines de référence L_ref
              Distance de séparation d_sep
    sortie :  Liste des streamlines correspondantes L_corres

L_corres := liste_vide
G := grille_vide
Pour chaque streamline de référence S_r dans L_ref
  Répète
    continue := Faux
    meilleurScore := 0
    Pour chaque sample point p de S_r
      S_c := IntègreStreamline(V_t, G, p, d_sep)
      Si S_c est valide
        continue := Vrai
        Evaluer le score de corrélation de S_c par rapport à S_r
        Si score > meilleurScore
          meilleureCandidate := S_c
          meilleurScore := score
        FinSi
      FinSi
    FinPour
    Si continue := Vrai
      Ajouter meilleureCandidate dans L_corres
      Référencer tous les sample point de meilleurCandidate dans G
    FinSi
  Jusqu'à continue := Faux
FinPour
Retourne L_corres
```

2.3 Insertion de nouvelles streamlines

La seconde phase de notre algorithme consiste à compléter l'ensemble des streamlines correspondantes avec de nouvelles streamlines afin d'atteindre la densité souhaitée. Cette phase est nécessaire car la sélection des streamlines correspondantes garantit par construction que l'on ne dépasse pas la densité souhaitée, mais elle n'assure pas une couverture complète et uniforme de l'image. Ainsi, il est possible qu'à la fin de la première phase, particulièrement lors de changements importants dans la structure du flux, quelques régions de l'image n'aient plus la densité de streamlines désirée.

Pour compléter l'image nous utilisons l'algorithme de placement décrit dans la section 2.4 du chapitre 3. L'ensemble initial de streamlines qu'utilise la fonction **`PlaceStreamlines`** est constitué des streamlines construites lors de la première phase. Rappelons que **`PlaceStreamlines`** a pour effet de placer de nouvelles streamlines par propagation à partir des streamlines initiales et des streamlines qui pourraient être dérivées. La fonction s'arrête quand aucune nouvelle streamline ne peut être insérée, c'est à dire quand la densité souhaitée a été atteinte. Les nouvelles streamlines créées lors de cette phase sont insérées à la suite des streamlines déjà placées, afin que les futures streamlines correspondantes soient construites sous les mêmes contraintes que l'ensemble courant.

Nous avons dit dans l'introduction de la section 2 que l'insertion de nouvelles streamlines, non corrélées avec les streamlines du pas de temps précédent, était la source la plus importante de dégradation de la qualité de l'animation. Nous pourrions relâcher la contrainte sur le critère de densité au profit de la qualité de l'animation en supprimant cette phase de remplissage par de nouvelles streamlines. Malheureusement, il en résulterait une diminution progressive du taux de couverture de l'image, laissant des régions de taille croissante sans information sur la direction du flux. De plus la dégradation de l'animation induite par l'introduction de ces nouvelles streamlines est limitée du fait du faible nombre de streamlines qu'il est possible d'insérer à chaque pas de temps. En effet les streamlines correspondantes encombrent déjà fortement le domaine et ne laisse que peu de place à de nouvelles streamlines pour croître et atteindre la taille minimum requise pour qu'elles soient valides. En pratique, les nouvelles streamlines sont peu nombreuses et sont généralement courtes. Il en résulte que le nombre de pixels couverts par de nouvelles streamlines est relativement faible. L'effet de l'apparition de nouvelles streamlines est ainsi minimisé. De plus, nous proposons dans la section 2.4.4 une technique permettant d'atténuer l'effet visuel provoqué par l'apparition de ces nouvelles streamlines.

2.4 Amélioration de la qualité de l'animation

Bien que l'algorithme de sélection des streamlines correspondantes donne des animations de bonne qualité grâce à la bonne corrélation des ensembles de streamlines, il subsiste quelques effets d'aliassage. Nous proposons quelques techniques pour améliorer la fluidité de l'animation.

2.4.1 Interpolation de champs de vecteurs intermédiaires

Si l'échantillonnage temporel des champs de vecteurs est trop faible au regard de la dynamique du phénomène qu'ils représentent, les structures des champs de vecteurs consécutifs seront très différentes. Il en résulte une forte variation des formes des streamlines d'une image à l'autre de l'animation, ce qui a tendance à limiter la compréhension de l'évolution de la structure du flux. Il est possible d'adoucir les transitions entre les mesures de l'échantillonnage en insérant des champs de vecteurs interpolés entre ces mesures. Les streamlines sembleront alors se déformer progressivement d'une image à l'autre facilitant ainsi le suivi de l'évolution de la structure du flux. Techniquement, chaque champ de vecteurs intermédiaire est généré à la volée, avant la construction des streamlines, en faisant une moyenne pondérée des vecteurs se trouvant dans deux champs de mesure consécutifs. La mise en place de cette technique requiert le stockage de trois champs de vecteurs en mémoire centrale (deux champs de vecteurs consécutifs plus le champ de vecteurs intermédiaire) en comparaison avec le seul champ de vecteurs pour le pas de temps courant quand l'interpolation n'est pas demandée.

2.4.2 Priorité aux streamlines circulaires

Les streamlines circulaires sont des objets importants dans les représentations des champs de vecteurs car elles montrent l'emplacement de points critiques dont l'évolution est généralement intéressante pour la compréhension de la dynamique du phénomène observé. Par exemple en météorologie, les streamlines circulaires peuvent représenter le centre d'anticyclones ou de dépressions atmosphériques. Pour faciliter le suivi de l'évolution de ces caractéristiques au cours de l'animation, il est important d'éviter qu'elles ne disparaissent et apparaissent de manière incontrôlées. Il est plutôt souhaitable de faire évoluer leur forme et leur position au cours de l'animation. Pour cette raison, nous modifions l'ordre de construction des streamlines en plaçant les streamlines circulaires en tête de la liste des streamlines de référence. Toutes les streamlines circulaires sont détectées durant l'intégration si leurs extrémités se rapprochent à une distance inférieure à un certain seuil (voir section 2.2 du chapitre 3). A la fin de la seconde phase de notre algorithme, toutes les streamlines circulaires sont triées par ordre de longueur croissante (pour placer en premier les plus petites) et insérées en début de la liste des streamlines.

2.4.3 Effet *Tapering*

Dans l'animation de champs de vecteurs non stationnaires, les extrémités des streamlines ne se trouvent pas nécessairement au même endroit pour des streamlines correspondantes consécutives. Le «mouvement» de ces extrémités à tendance à introduire un effet de clignotement et accroche l'oeil de l'observateur. La diminution progressive de l'épaisseur des extrémités des streamlines (appelé effet *tapering*) a pour conséquence de réduire la perception de ces extrémités et ainsi de limiter l'effet visuel indésirable. La technique utilisée est la même que celle proposée dans la section 4.1.1 du chapitre 3 pour améliorer la qualité visuelle des représentations éparses à base de streamlines.

2.4.4 Epaississement progressif des streamlines

De façon à atténuer l'effet visuel dû à l'apparition de nouvelles streamlines non corrélées au cours de l'animation, l'épaisseur des streamlines est fonction de leur *âge*. Une nouvelle streamline est créée avec une épaisseur de 1 et cette épaisseur croît d'un pas de temps à l'autre pour ses streamlines correspondantes jusqu'à atteindre l'*épaisseur adulte*.

2.4.5 Effet de traîne

Comme pour la visualisation de petits objets, telles que des particules, pour lesquelles un effet de traîne est utilisé de façon à faciliter leur suivi d'une image à l'autre, un effet de traîne sur les streamlines permet d'augmenter leur corrélation entre les images successives de l'animation. Notre expérience a montré que cet effet de rémanence augmente considérablement le suivi des caractéristiques du flux au cours de l'animation. A chaque pas de temps, les ensembles de streamlines calculés pour plusieurs pas de temps consécutifs sont dessinés dans la même image, chaque streamline ayant une intensité aui est fonction de l'instant auquel elle a été construite (les streamlines les plus récentes ayant des intensités plus importantes). Quand l'effet de traîne est utilisé, il est important de remarquer que des streamlines dans l'image peuvent se couper si elles ont été calculées à des instants différents (voir figure 5.7).

3 Corrélation des textures des streamlines

Comme pour les flux stationnaires, des dégradés de couleurs sont plaqués sur les streamlines pour visualiser l'orientation du flux. La perception de mouvement est produite en déplaçant ces dégradés dans le sens du flux le long des streamlines, d'une image à l'autre de l'animation. Bien que l'idée du déplacement des dégradés soit la même que pour le cas des flux stationnaires, la réalisation a du être totalement repensée de façon à s'adapter à la nature des flux non-stationnaires.

Rappelons que dans le cas stationnaire, la structure du flux est invariante au cours du temps et l'animation est produite à partir d'un seul ensemble de streamlines. Le plaquage des dégradés sur les streamlines n'est fait qu'une seule fois pour la première image, les images suivantes sont obtenues par une opération de décalage sur la table des couleurs. Enfin l'allocation des couleurs des dégradés est incrémentale, il suffit de choisir aléatoirement la couleur d'un sample point d'origine et déduire la couleur des sample points suivants sur la streamline en ajoutant un incrément dépendant de la vitesse instantanée du flux en ce point.

Dans le cas des flux non stationnaires, la structure du flux varie d'une image à l'autre. Il est donc nécessaire de recalculer le plaquage des dégradés sur les streamlines à chaque pas de temps. L'impression de mouvement sera de bonne qualité si les dégradés sur toute portion de streamline semble se déplacer dans le sens du flux sur la streamline correspondante. Le mode d'allocation des dégradés doit maintenant tenir compte de la déformation des streamlines, de leur changement de longueur et de leur déplacement dans le domaine. Comme l'allocation incrémentale des dégradés décrite pour les flux stationnaires ne prend pas en compte ces contraintes, elle ne permet pas de corréler les dégradés tout le long des streamlines correspondantes. Le mouvement du flux n'est alors pas interprétable. Pour optimiser la corrélation des dégradés d'une streamline avec ceux de sa streamline de référence, nous avons mis au point un nouveau mode d'allocation de ces dégradés. L'idée est de décomposer chaque streamline en plusieurs portions sur lesquelles seront plaquées les dégradés. La corrélation des dégradés revient alors à corréler les portions des streamlines avec les portions des streamlines de références.

Les dégradés que nous utilisons ici sont stockés dans une texture 1D. Les portions de streamlines sur lesquelles sont plaqués les dégradés sont appelées les *supports de texture*. La section suivante décrit la façon dont les supports de texture sont répartis le long des streamlines alors que la section 3.2 montre comment la texture est plaquée sur ces supports pour faire apparaître l'impression de mouvement.

3.1 Corrélation des supports de texture

Plus les supports de texture d'une streamline seront corrélés avec ceux de sa streamline de référence et plus l'impression de mouvement du flux sera bonne. Paradoxalement, nous n'allons pas chercher à mettre les supports de texture en mouvement les uns par rapport aux autres, mais au contraire, nous chercherons à ce qu'ils restent le plus immobile possible d'une image à l'autre de l'animation. L'effet de mouvement sera alors donné ultérieurement en utilisant les possibilités du plaquage de textures 1D (voir section 3.2). Plus les supports sembleront fixes d'une image à l'autre de l'animation et plus on dira que leur corrélation est élevée.

Les supports de textures d'une streamline sont définis par leurs extrémités et leur longueur. Les extrémités des supports sont des sample points de la streamline appelés *points de contrôle* et leur longueur est le nombre de sample points entre deux points de contrôle. Corréler les supports de textures revient alors à corréler les points de contrôle d'une streamline avec ceux de sa streamline de référence. Les points de contrôle sont déterminés en deux phases. Dans une première phase on recherche tous les points de contrôle qu'il est possible de déduire des *points de contrôle de référence* (se trouvant sur les streamlines de référence), ensuite, dans une deuxième phase, on homogénéise la longueur des supports en ajoutant ou en supprimant des points de contrôle sur les streamlines.

Pour le premier ensemble de streamlines, il n'existe pas de points de contrôle de référence. Les points de contrôle sont donc placés sur les streamlines de telle sorte que la longueur des supports soit d'une taille souhaitée $n_{control}$ (par exemple $n_{control} = 10$ ou 15 sample points). Pour les images suivantes, pour chaque streamline on détermine les *points de contrôle correspondants* en choisissant les sample points les plus proches des points de contrôle de référence. De façon à ce que les extrémités des supports ne varie pas trop en position d'une image à l'autre, seuls les sample points se trouvant dans le voisinage du point de contrôle de référence sont pris en compte dans la recherche du point de contrôle correspondant. Si une streamline correspondante s'écarte trop par endroit de sa streamline de référence, il est possible que des points de contrôle de référence ne trouvent pas de correspondant (voir figure 5.5). La grille de proximité évoquée dans la section 2.1.3 du chapitre 3 est utilisée afin d'accélérer la recherche des sample points les plus proches des points de contrôle.

Une deuxième phase est maintenant nécessaire pour finir de répartir les points de contrôle sur les streamlines. En effet, les nouvelles streamlines insérées au pas de temps courant ne possèdent pas de streamline de référence et ne porte alors pour le moment aucun point de contrôle. De plus, la longueur des supports sur les autres streamlines risque d'être très variable : elle sera longue si un ou plusieurs points de contrôle de référence n'ont pas trouvé de correspondants ou

courte si la forme de la streamline fait que deux points de contrôle consécutifs se sont rapprochés. Pour ne pas que la texture qui sera plaquée sur ces supports ne soient trop étirées ou contractées, cette seconde phase permettra d'homogénéiser la longueur des supports autour de la longueur souhaitée $n_{control}$. Nous avons défini $min\_n_{control}$ et $max\_n_{control}$ comme la longueur minimum et maximum autorisée pour les supports respectivement. L'ordre des opérations est le suivant :

- parcourir toutes les streamlines qui possèdent déjà des points de contrôle,
  - si la longueur d'un support est inférieure à $min\_n_{control}$, supprimer le point de contrôle commun avec le support adjacent le plus court.
  - si la longueur d'un support est supérieure à $max\_n_{control}$, diviser ce support en ajoutant des points de contrôle intermédiaires.
- parcourir toutes les nouvelles streamlines et placer des points de contrôle tous les $n_{control}$ sample points.

En pratique on fixe $min\_n_{control} = 0.5 \times n_{control}$ et $max\_n_{control} = 1.5 \times n_{control}$.

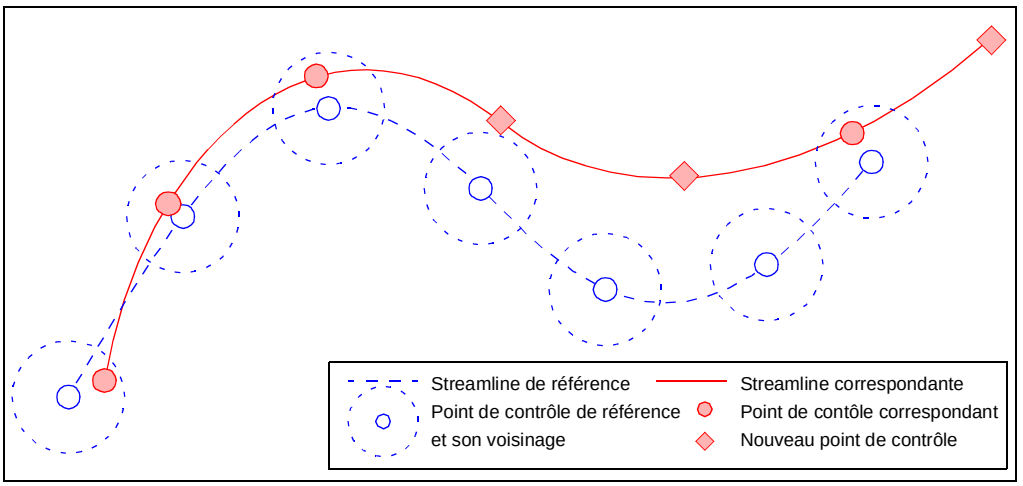

Figure 5.5 Répartition des points de contrôle sur les streamlines.

La figure 5.5 montre une streamline de référence, ses points de contrôle et leur voisinage, et une streamline correspondante avec ses points de contrôle. Sur la streamline correspondante, les cercles ont été trouvés à partir des points de contrôle de référence alors que les losanges ont été créés lors de la deuxième phase. Il est à noter que l'insertion ou la suppression de points de contrôle réduisent la corrélation des supports de texture mais ces modifications sont nécessaires pour garder une bonne visibilité des textures qui seront plaquées par la suite ; de plus, les modifications sur les streamlines sont locales et ne remettent pas en cause la corrélation sur toute leur longueur.

3.2 Mouvement des textures le long des streamlines

L'objectif de cette partie est de plaquer des textures sur les supports de texture des streamlines de façon à ce qu'une impression de mouvement dans le sens du flux apparaisse au cours de l'animation. Nous sommes amenés dans un premier temps à expliquer brièvement le mécanisme de plaquage de texture 1D sur une ligne brisée et à montrer dans un deuxième temps la façon dont nous avons utilisé ce mécanisme pour obtenir un effet de déplacement des textures le long des streamlines.

3.2.1 Plaquage de texture 1D sur une ligne brisée

Une texture 1D est une série de couleurs. Le plaquage d'une texture 1D sur un segment consiste à allouer à chaque pixel du segment une couleur de la texture. Le plus souvent la texture est stockée dans un tableau de couleurs. Le mécanisme de plaquage prend en charge le choix de la couleur à affecter à chaque pixel du segment lors de son tracé. Si le nombre de pixels du segment est supérieur au nombre de cellules dans le tableau de la texture, la texture est dite *étirée* sur le segment, s'il est inférieur on dit que la texture est *contractée* sur le segment.

Il n'est pas nécessaire que la totalité de la texture soit représentée sur le segment. Pour spécifier la partie de la texture qui sera plaquée sur le segment, on précise une *coordonnée de texture* à chaque extrémité du segment. Ainsi, seules les couleurs de la texture se trouvant dans l'intervalle spécifié par les coordonnées de texture seront allouées aux pixels du segment. Généralement, dans les implémentations des librairies graphiques (telles que OpenGL ou MesaGL) la longueur d'une texture 1D est ramenée à l'unité. Ainsi, la totalité de la texture sera plaquée sur le segment si une de ses extrémités possède une coordonnée de texture égale à 0 et l'autre égale à 1. A titre d'exemple, la deuxième moitié de la texture sera plaquée sur le segment si ses extrémités ont des coordonnées de texture égales à 0.5 et 1. Il est possible de répéter plusieurs fois une texture sur un même segment si la différence des coordonnées de textures est supérieure à 1. Ainsi pour un segment ayant des coordonnées de texture égales à 0 et 2.5, la texture sera reproduite 2,5 fois sur le segment.

Pour couvrir une ligne brisée avec la totalité d'une texture il suffit d'allouer les coordonnées 0 et 1 (ou tout autre paire de coordonnées dont la différence est 1) aux extrémités de la ligne brisée et des coordonnées intermédiaires à chaque autre point. Les coordonnées doivent être monotones (soient croissantes, soient décroissantes) le long de la ligne brisée afin que la texture plaquée ait la même allure que la texture initiale. La texture sera reproduite *n* fois sur la ligne brisée si la différence entre les coordonnées des extrémités est égale à *n* et que les coordonnées intermédiaires sont monotones.

Pour obtenir un effet de déplacement de la texture le long de cette ligne brisée il suffit d'ajouter un incrément à la coordonnée de texture de chaque point à

chaque pas de temps de l'animation. Si l'incrément $incr_{anim}$ est le même pour tous les points, le nombre de textures apparentes sur la ligne brisée sera constant. L'effet de déplacement apparaît parce qu'à chaque pas, les textures sont décalées sur la ligne brisée. C'est la valeur de l'incrément $incr_{anim}$ qui fixe la vitesse apparente de déplacement de la texture. La vitesse apparente de déplacement est minimum pour une valeur de $incr_{anim}$ proche de 0 et maximum pour une valeur proche de 0.5. Toute valeur comprise entre 0.5 et 1 produira un effet de déplacement négatif.

3.2.2 Application aux streamlines

L'application du plaquage de textures 1D sur les streamlines est relativement directe. Les streamlines sont des lignes brisées sur lesquelles nous avons différencié certains points que nous avons appelé points de contrôle. Ces points de contrôle définissent les supports sur lesquels sera plaquée la totalité de la texture (c'est à dire que la différence des coordonnées de texture de deux points de contrôle consécutifs sera égale à 1). La coordonnée de texture de chaque sample point appartenant à un support particulier est déterminée en fonction de son rang sur le support, de la longueur du support et de l'incrément $incr_{anim}$. L'algorithme suivant montre comment sont allouées les coordonnées de texture sur tous les sample points des streamlines pour une image donnée.

```
Fonction AllouerCoordonneesTexture(L, num_image)
     entrée :  Liste des streamlines L dans l'image courante
               Numéro de l'image courante num_image
               (comme paramètre global : incr_anim )

incr_image = incr_anim * num_image
Pour chaque streamline S dans L
  | Pour chaque point de contrôle pc sur S
  |   | Si pc est le premier point de contrôle de S
  |   |   | pc1 = pc
  |   | Sinon
  |   |   | longueur = rang(pc) - rang(pc1)
  |   |   | increment = 1.0 / longueur
  |   |   | incr_sp = increment
  |   |   | Pour chaque sample point sp allant de pc1 (non compris) à pc
  |   |   |   | sp.coord_texture = pc1.coord_texture - incr_sp - incr_image
  |   |   |   | incr_sp = incr_sp + increment
  |   |   | FinPour
  |   |   | pc1 = pc
  |   | FinSi
  | FinPour
FinPour
```

Où *num_image* est le numéro de l'image courante, la fonction **rang(***sp***)** renvoie la position du sample point *sp* sur sa streamline et $sp.coord_{texture}$ correspond à la coordonnée de texture du sample point. Dans l'algorithme, on voit clairement que la coordonnée de texture d'un sample point dépend de la coordonnée de texture du premier point de contrôle du support, d'un incrément dû à son rang sur le support et d'un incrément dû à l'image dans lequel se

trouve le sample point. On remarque que les incréments sont soustraits à la coordonnée de textures des sample points. Ce choix est justifié par le sens de déplacement que l'on souhaite appliquer aux textures qui est le sens du flux.

4 Résultats et discussions

Nous présentons dans cette section des résultats obtenus dans les différents mode de représentation que notre algorithme autorise. Ces représentations vont d'un placement éparse des streamlines à une répartition dense, sans textures pour ne visualiser que la déformation de la structure du flux ou avec textures de façon à en observer l'orientation. La suite de la section concerne quelques améliorations ou limitations sur notre approche ou l'état actuel de nos développements.

Dans un premier temps voici un éventail des diverses représentations des flux non stationnaires que permet notre approche. Les données visualisées sont des courants atmosphériques au dessus de l'Europe. La figure 5.6 montre deux images de même rang dans deux animations d'un même champ de vecteurs. Sur celle de gauche, seule la déformation de la structure des vents est visible. Le déplacement des streamlines circulaires informe sur l'évolution du centre des dépressions et anticyclones. Sur l'image de droite, l'orientation des vents est visualisé par le plaquage de textures sur les streamlines. Les motifs se déplacent le long des streamlines au cours de l'animation.

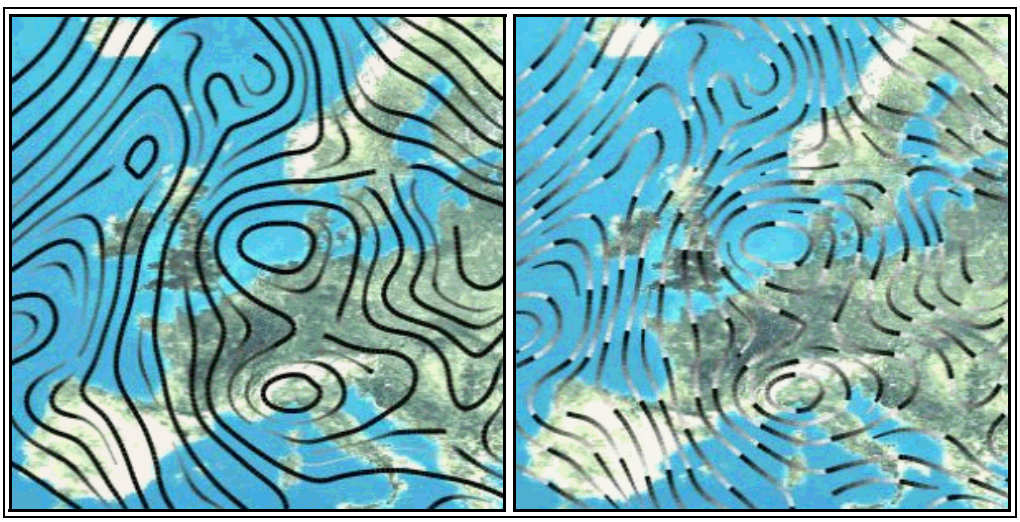

Figure 5.6 Evolution de la structure des courants atmosphériques au dessus de l'Europe. Les streamlines semblent se déformer au cours du temps.

Pour la figure 5.7, un effet de traine à été utilisé pour faciliter le suivi des streamlines. Ici, pour chaque image de l'animation, les streamlines des trois derniers pas de temps ont été affichées. Cette technique augmente la corrélation d'une image à l'autre en conservant les deux tiers des streamlines invariantes.

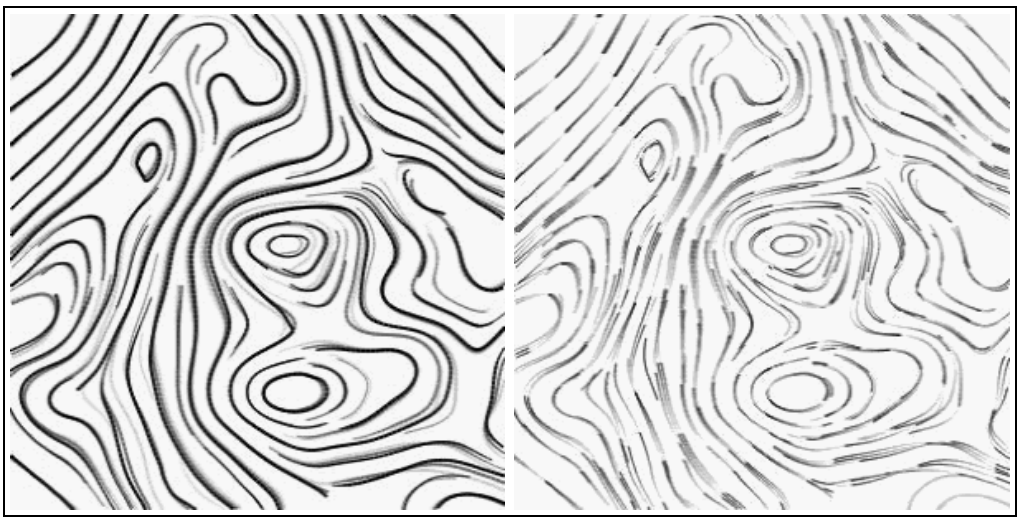

Figure 5.7 Effet de traine. (a) sans et (b) avec plaquage de texture.

Les images de la figure 5.8 sont des représentations denses du même champ de vecteurs. Elles ont été obtenues en choisissant une distance de séparation entre streamlines suffisamment petite. En plus de visualiser l'orientation, le plaquage de texture permet de différencier les streamlines les unes des autres.

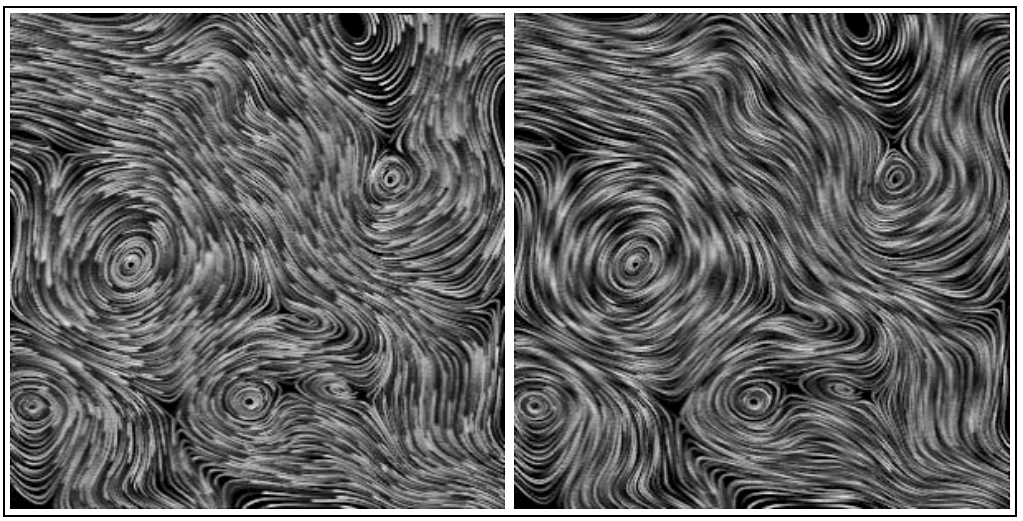

Figure 5.8 Représentation dense de champs de vecteurs non stationnaires. (a) avec une texture de type "dent de scie" et (b) avec une texture de type "sinusoidal".

Remplacement des textures 1D par des icônes pour les représentations éparses

De même que pour le rendu visuel de streamlines dans la section 4.1.2 du chapitre 3, les supports de textures peuvent être utilisés pour placer des icônes le long des streamlines. Si les icônes sont orientées il est alors possible de visualiser l'orientation du flux. Notons que si on se contente de placer une icône sur chaque support, les icônes sembleront attachées aux streamlines et aucun mouvement relatif le long des streamlines ne sera visible puisque les supports ont été calculés de façon à justement minimiser leur déplacement. Il est alors nécessaire de placer les extrémités de chaque icônes sur des sample points particuliers appartenant au supports. Le déplacement de l'emplacement des extrémités des icônes sur les supports donnera l'effet de mouvement des icônes le long des streamlines durant l'animation. Remarquons tout de même que d'important effets d'aliassage ou de clignotement risque de survenir lors de la disparition et de l'apparition d'icônes d'une image à l'autre. Ces disparitions et apparitions surviennent dès qu'un support n'a pas de correspondant dans l'image suivante ou qu'une nouvelle streamline est insérée dans la représentation. Ces cas se produisent de la même manière pour les textures mais la disparition ou apparition d'un segment coloré trouble moins l'animation qu'une icône visuellement plus contrastée. Il est alors impératif, pour améliorer la qualité de l'animation de procéder à une phase de post-traitement visant à minimiser le clignotement des icônes. Cette phase pourrait consister à ne retenir que les icônes apparaissant dans un nombre minimum d'images successives et faire croître (resp. décroître) l'intensité (ou l'opacité) des icônes lors de leur apparition (resp. jusqu'à leur disparition).

Pas de vitesse variable

Au stade actuel de nos travaux, le déplacement des textures le long des streamlines s'effectue à vitesse constante. Il n'est pour l'instant pas possible de rendre compte des différentes vitesses du flux dans le domaine comme nous l'avons réalisé dans la Motion Map pour les flux stationnaires. L'idée serait d'étirer la texture sur la streamline aux endroits où la vitesse instantanée est grande et de la contracter dans les zones de faible vitesse. Malheureusement, notre concept de supports de texture impose de plaquer une texture complète entre deux points de contrôle ce qui limite la possibilité d'étirement ou de contraction des textures. Il semble que l'ajout de vitesse variable dans la représentation des flux non stationnaires nécessite l'imagination d'un autre mode de corrélation des textures.

Interprétation et limitation des représentations denses texturées

Dans le cas des représentation éparses, l'information principale qu'un observateur retire en visualisant nos animations est la «déformation» des streamlines au cours du temps. L'information complémentaire qui lui est offerte

est l'observation de l'orientation du flux par le déplacement des motifs des textures sur les streamlines.

Dans les représentations denses, les streamlines n'apparaissent plus distinctement et le seul mouvement discernable est le déplacement des motifs que forment les textures. L'information principale n'est plus l'évolution de la structure du flux mais le mouvement global du flux. Un piège pour l'observateur est qu'il est maintenant tentant d'assimiler le déplacement de ces motifs au cours du temps à des particules dérivant au gré du flux. Or il faut rappeler que la trajectoire de telles particules serait des pathlines et non les portions de streamlines sur lesquelles évoluent les motifs. Bien que l'évolution de la structure du flux soit parfaitement interprétable, il est important d'avoir conscience que le mouvement apparent de nos représentations est une approximation du mouvement réel du flux.

5 Conclusion

Dans ce chapitre, nous nous sommes intéressés à la visualisation de champs de vecteurs bidimensionnels non stationnaires. Nous proposons d'animer les états instantanés successifs des champs de vecteurs, représentés par des ensembles de streamlines, de façon à faire apparaître l'évolution de la structure du flux au cours du temps. Une des difficultés à résoudre a été de corréler les ensembles consécutifs de streamlines afin d'obtenir une déformation continue et fluide de la structure. Un mécanisme original de corrélation de textures sur les streamlines permet aussi de visualiser l'orientation du flux en déplaçant des motifs colorés. Malgré la variation en position, forme et longueur des streamlines successives, nous avons réussi à produire des animations très fluides quelle que soit la densité. Le même algorithme nous permet de produire des représentations de densité quelconque en fixant la distance de séparation entre les streamlines.

CHAPITRE

6

Conclusion

Au cours de cette thèse, nous avons développé des algorithmes pour visualiser des champs de vecteurs bidimensionnels. Nos travaux sont basés sur le placement et la coloration de courbes particulières, appelées «lignes de courant» ou encore «*streamlines*». Un ensemble de streamlines réparties dans le domaine permettent de visualiser la structure du flux en révélant l'information directionnelle ainsi que l'emplacement des points critiques.

Nous avons développé un algorithme de placement autorisant un contrôle précis de la densité de streamlines dans le domaine. Il nous est ainsi possible de produire des ensembles uniformes de streamlines de densité quelconque, allant de éparse à dense. La densité est fixée en spécifiant simplement la distance de séparation minimale entre les streamlines. Une partie importante de l'algorithme de placement concerne la sélection de leur points de départ (seed points). Parmi les techniques de sélection que nous avons développées, la méthode de sélection dans le voisinage des streamlines déjà placées s'est montré le plus efficace. Notre algorithme s'avère être environ 30 fois plus rapide que celui de Turk et Banks, qui fait office de méthode de référence pour ce problème dans la littérature. Une extension immédiate de l'algorithme permet de produire des ensembles de streamlines multirésolution pour lesquels nous avons développé un système de navigation interactive. Nous nous sommes aussi intéressés aux modes de rendu visuels des streamlines aux différentes densités. Ainsi, l'utilisation de fonctions d'intensité pour colorer les streamlines nous a permis d'obtenir des représentations denses statiques proches de celle produites par d'autres techniques de visualisation telles que *LIC* ou *SpotNoise*.

Afin de visualiser l'orientation du flux de façon naturelle, nous avons remplacé les fonctions d'intensité par un mode d'allocation de couleurs qui permet d'animer le flux à l'aide de la technique d'animation de la palette des couleurs. Nous nous sommes particulièrement intéressés aux représentations denses qui offrent une résolution spatiale maximale. Afin d'accélérer le calcul des représentations nous avons adapté notre algorithme de placement de streamlines au cas dense. La simplification du test de distance a permis de gagner un facteur 10 en temps de calcul. Une caractéristique intéressante de

notre méthode est que toute l'information nécessaire à l'animation de *N* images peut être stockée dans une structure de donnée originale ayant la taille d'une seule image, que nous avons appelé *Motion Map*. Une mesure de la corrélation basée sur la longueur des streamlines nous a permis d'affiner l'algorithme de discrétisation des streamlines dans la Motion Map afin de produire des animations de haute qualité. Notre méthode permet de représenter les différentes vitesses locales du champ de vecteurs dans des animations parfaitement cycliques. Notons encore que l'animation ne requiert que peu de ressources quand elle est visualisée avec la technique d'animation de la palette des couleurs, ce qui permet d'obtenir des animations temps réel, y compris sur les équipements informatiques d'entrée de gamme. Une autre façon d'obtenir une animation à partir de la Motion Map consiste à dériver un ensemble cyclique de textures qui peuvent être plaquées successivement sur les objets d'une scène 3D.

De façon à visualiser les champs de vecteurs non stationnaires, nous avons dû tenir compte de leur évolution temporelle. Pour cela, nous avons développé deux nouvelles méthodes afin de corréler la structure, constituée d'un ensemble de streamlines, et l'orientation, visualisée par le déplacement de textures. Pour corréler les ensembles successifs de streamlines, nous nous sommes dotés d'un critère de distance entre deux streamlines. Ce critère relatif permet de sélectionner parmi un ensemble de streamlines candidates calculé au temps *t*, celle qui ressemble le plus, de par sa position et sa forme, à une streamline de référence calculée au temps *t-1*. Du fait du changement de forme et de longueur des streamlines d'un pas de temps à l'autre, nous avons mis au point un nouveau mode d'allocation de couleurs que pour la Motion Map. Celui-ci est basé sur le déplacement de textures 1D le long des streamlines. Afin de maintenir la meilleure corrélation possible, les supports de textures sont localement compressés ou dilatés, supprimés ou insérés sur les streamlines. Les animations produites sont très fluides et permettent un suivi précis de l'évolution temporelle des champs de vecteurs non stationnaires. Le contrôle de la densité de streamlines dans les images successives permet de produire des représentations tant éparses que denses.

La prochaine étape importante en visualisation de champs de vecteurs concerne la visualisation de champs tridimensionnels qui a encore été peu abordée à ce jour, même si une adaptation de notre algorithme de placement au cas 3D a déjà été réalisée par Furhmann [20]. Le passage aux champs tridimensionnels permettra à ce domaine de recherche d'évoluer vers sa maturité en autorisant la visualisation de phénomènes physiques qui sont, par essence, tridimensionnels et non stationnaires. J'ai l'opportunité de continuer à participer à cette recherche motivante dans le cadre d'un postdoctorat à l'Université d'Etat de Floride (USA).

Bibliographie

[1] Arvo, J. et D. Kirk. An Introduction to Ray Tracing. A Glassner Ed, pages 217-227. Academic Press.

[2] Banks, D. Illumination in Diverse Codimensions. Dans Computer Graphics (*Proceedings de SIGGRAPH'94*), pages 327-334. ACM Press.

[3] Battke, H., D. Stalling et H-C. Hege. Fast Line Integral Convolution for Arbitrary Surfaces in 3D. Dans H-C. Hege et K. Polthier, Editeurs, *Visualization and Mathemathics*, pages 181-195, 1997

[4] Becker, B., D. Lane et N. Max. Unsteady Flow Volumes. Dans G. Nielson et D. Silver, Editeurs (*Proceedings de IEEE Visualization'95*, 29 Octobre - 3 Novembre, 1995), pages 329-335, IEEE Press.

[5] Brill M., H. Hagen, H-C. Rodrian, W. Djatschin et S. Klimenko. Streamball Techniques for Flow Visualization. Dans R.D. Bergeron et A. Kaufman, Editeurs (*Proceedings de IEEE Visualization'94*, 17-21 Octobre, 1994), pages 225-231, IEEE Press.

[6] Buning, P. Sources of Error in the Graphical Analysis of CFD results. *Journal of Scientific Computing*, 3(2), 1988, pages 149-164.

[7] Burden, R.L. et J.D. Faires. Numerical Analysis, sixth edition. Brooks/Cole Publishing Company.

[8] Cabral, B. et L. Leedom. Imaging Vector Fields Using Line Convolution. Dans Computer Graphics (*Proceedings de SIGGRAPH'93)*, pages 263-272. ACM Press.

[9] Canuto, C., M.Y. Hussaini, A. Quarteroni et T.A. Zang. Spectral Method in Fluid Dynamics. *Springer Series in Computational Physics*. Springer Verlag.

[10] Chédot, C. et W. Lefer. Multi-modal Flow Animation with the Motion Map. *Late Breaking Hot Topics, IEEE Visualization'98*, 18-23 Octobre, 1998.

[11] Computational Fluid Dynamics. G. de Vahl Davis et C. Fletchs, Editeurs (*Proceedings de the International Symposium on CFD*, Sydney, Australia, Août 1987). North Holland.

[12] de Leeuw, W.C. et J.J. van Wijk. A Probe for Local Flow Field Visualization. Dans G.M. Nielson et R. Dan Bergeron, Editeurs (*Proceedings de IEEE Visualization'93*, 25-29 Octobre, 1993), pages 39-45, IEEE Press.

[13] de Leeuw, W.C. et J.J. Van Wijk. Enhanced Spot Noise for Vector Field Visualization. Dans G. Nielson et D. Silver, Editeurs (*Proceedings de IEEE Visualization'95*, 29 Octobre - 3 Novembre, 1995), pages 233-239, IEEE Press.

[14] de Leeuw, W.C. Presentation and Exploration of Flow Data. *Thèse de doctorat à l'université de Delft*, Pays Bas, Juin 1997.

[15] de Leeuw, W.C. et R. van Liere. Comparing LIC and Spot Noise. Dans H. Hagen et H. Rushmeier, Editeurs (*Proceedings de IEEE Visualization'98*, 18-23 Octobre, 1998), pages 359-366, IEEE Press.

[16] Dovey D. Vector Plots for Irregular Grids. Dans G. Nielson et D. Silver, Editeurs (*Proceedings de IEEE Visualization'95*, 29 Octobre - 3 Novembre, 1995), pages 233-239, IEEE Press.

[17] Forsell, L.K. Visualizing Flow Over Curvilinear Grid Surfaces Using Line Integral Convolution. Dans R.D. Bergeron et A. Kaufman, Editeurs (*Proceedings de IEEE Visualization'94*, 17-21 Octobre, 1994), pages 240-247, IEEE Press.

[18] Forsell, L.K. et 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), pages 133-141, 1995.

[19] Freeman, W.T., E.H. Adelson et D.J. Heeger. Motion Without Movement. Computer Graphics 25(4) (*Proccedings de SIGGRAPH'91*, 28 Juillet - 2 Août, 1991), pages 27-30, ACM Press.

[20] Fuhrmann, A. et E. Gröller. Real-Time Techniques For 3D Flow Visualization. Dans H. Hagen et H. Rushmeier, Editeurs (*Proceedings de IEEE Visualization'98*, 18-23 Octobre, 1998), pages 305-312, IEEE Press.

[21] van Gelder, A. et J. Wilhelms. Interactive Animated Visualization of Flow Fields. Dans *proceedings de ACM Workshop on Volume Visualisation*, 1992, pages 47-54.

[22] Globus, A., C. Levit et T. Lasinski. A Tool for Visualizing the Topology of Three-Dimensional Vector Fields. Dans G.M. Nielson et L. Rosenblum, Editeurs (*Proceedings de Visualization'91*, 18-23 Octobre, 1991), pages 33-39, IEEE Press.

[23] Hearn D. et P. Baker. Computer Graphics - Second Edition. Prentice Hall International Editions, 1994.

[24] Hege, H.C., D. Stalling. LIC: Acceleration, Animation, and Zoom. Dans *Texture Synthesis with Line Integral Convolution (course notes)*, ACM SIGGRAPH'97, pages 17-49.

[25] Helman, J.L. et L. Hesselink. Representation and Display of Vector Field Topology in Fluid Flow Datasets. *IEEE Computer Graphics* 22(8), 1989, pages 27-36.

[26] Helman, J.L. et L. Hesselink. Surface Representations of two- and three-dimensional Fluid Flow Topology. Dans Arie Kaufman, Editeurs, (*Proceedings de Visualization'90*, 23-26 Octobre, 1990), pages 6-13, IEEE Press.

[27] Helman, J.L. et L. Hesselink. Visualizing Vector Field Topology in Fluid Flows. *IEEE Computer Graphics and Applications* 11(3), 1991, pages 36-46.

[28] Hairer, E., S.P. Nørsett et G. Wanner. Solving Ordinary Differential Equations I - Nonstiff Problems. Springer Verlag, 1993.

[29] Hultquist, J.P.M. Constructing Stream Surfaces in Steady 3D Vector Fields. Dans Arie E. Kaufman et Gregory M. Nielson, Editeurs (*Proceedings de IEEE Visualization'92*, 19-23 Octobre, 1992), pages 171-177, IEEE Press.

[30] Jobard, B. et W. Lefer. Creating Evenly-Spaced Streamlines of Arbitrary Density. Dans W. Lefer et M. Grave, Editeurs (*Proceedings du 8th Eurographics Workshop on Visualization in Scientific Computing'97*, 28-30 Avril, 1997), pages 43-55, Springer-Wien-NewYork.

[31] Jobard, B. et W. Lefer. The Motion Map: Efficient Computation of Steady Flow Animations. Dans R. Yagel et H. Hagen, Editeurs (*Proceedings de IEEE Visualization'97*, 19-24 Octobre, 1997), pages 323-328, IEEE Press.

[32] Johannsen, A. et R. Moorhead. Case Study: Visualization of Mesoscale Flow Features in Ocean Basins. Dans R.D. Bergeron et A. Kaufman, Editeurs (*Proceedings de IEEE Visualization'94*, 17-21 Octobre, 1994), pages 355-358, IEEE Press.

[33] Kenwright, D. et D. Lane. Optimization of Time-Dependent Particle Tracing Using Tetrahedral Decomposition. Dans G. Nielson et D. Silver, Editeurs (*Proceedings de IEEE Visualization'95*, 29 Octobre - 3 Novembre, 1995), pages 321-328, IEEE Press.

[34] Kiu, M-H. et D. Banks. Multi-Frequency Noise for LIC. Dans R. Yagel et G.M. Nielson, Editeurs (*Proceeding de Visualization'96*, 27 Octobre - 1 Novembre, 1996), pages 121-126, IEEE Press.

[35] Khouas L., C. Odet et D. Friboulet. Vector Field Visualization using Furlike Texture. Dans E. Gröller et H. Löffelmann et W. Ribarsky, Editeurs (*Proc. de Joint Eurographics/IEEE TCVG Symposium on Visualization*, Vienna, 26-28 Mai, 1999), pages 35-44, Springer-Wien-NewYork.

[36] Lane, D.A. Visualization of Time-Dependant Flow Fields. Dans G.M. Nielson et R. Dan Bergeron, Editeurs (*Proceedings de IEEE Visualization'93*, 25-29 Octobre, 1993), pages 32-38, IEEE Press.

[37] Lane, D.A. UFAT - A Particle Tracer for Time-Dependent Flow Fields. Dans R.D. Bergeron et A. Kaufman, Editeurs (*Proceedings de IEEE Visualization'94*, 17-21 Octobre, 1994), pages 257-264, IEEE Press.

[38] Lane, D.A. Visualizing Time-Varying Phenomena Dans Numerical Simulations of Unsteady Flows. *Technical report NAS-96-001*, 1996, http://www.nas.nasa.gov/Pubs/TechReports/NASreports/NAS_96_001/

[39] Löffelmann, H., L. Mroz, E. Gröller et W. Purgathofer. Hierarchical Streamarrows for the Visualization of Dynamical Systems. Dans Wilfrid Lefer et M. Grave, Editeurs (*Proceedings du 8th Eurographics Workshop on Visualization in Scientific Computing'97*, 28-30 Avril, 1997), pages 155-164, Springer-Wien-NewYork.

[40] Ma, K.L., P. Smith. Virtual Smoke: An Interactive 3D Flow Visualization Technique. Dans Arie E. Kaufman et Gregory M. Nielson, Editeurs (*Proceedings de IEEE Visualization'92*, 19-23 Octobre, 1992), pages 46-52, IEEE Press.

[41] Mao, X., M. Kikukawa, N. Fujita et A. Imamiya. Line Integral Convolution for 3D Surfaces. Dans W. Lefer et M. Grave, Editeurs (*Proceedings du 8th Eurographics Workshop on Visualization in Scientific Computing'97*, 28-30 Avril, 1997), pages 57-69, Springer-Wien-NewYork.

[42] Mao, X., Y. Hatanaka, H. Higashida et A. Imamiya. Image-Guided Streamline Placement on Curvilinear Grid Surfaces. Dans D. Ebert, H. Rushmeier et H. Hagen, Editeurs (*Proceedings de IEEE Visualization'98*, 18-23 Octobre, 1998), pages 135-142, IEEE Press.

[43] Max, N., B. Becker et R. Crawfis. Flow Volumes for Interactive Vector Field Visualization. Dans G.M. Nielson et R. Dan Bergeron, Editeurs (*Proceedings de IEEE Visualization'93*, 25-29 Octobre, 1993), pages 19-24, IEEE Press.

[44] Max, N., R. Crawfis et C. Grant. Visualizing 3D Velocity Fields Near Contour Surfaces. Dans R.D. Bergeron et A. Kaufman, Editeurs (*Proceedings de IEEE Visualization'94*, 17-21 Octobre, 1994), pages 348-354, IEEE Press.

[45] Max, N. et B. Becker. Flow Visualization Using Moving Textures. Dans *Proceedings de ICASE/LaRC Symposium on Visualizing Time-Varying Data*, 1995.

[46] Okada, A., D. Lane. Enhanced Line Integral Convolution with Flow Feature Detection. *NAS Technical Report NAS-96-007*, Juin 96, http://www.nas.nasa.gov/NAS/TechReports/NASreports/NAS-96-007/nas_96_007.html

[47] Post, F. H. et T. van Walsum. Fluid Flow Visualization. *Focus on Scientific Visualization. Springer*, 1993, pages 1-40.

[48] Sabella, P. A Rendering Algorithm for Visualizing 3D Scalar Fields. *IEEE Computer Graphics*, Vol 22, No 4, 1998, pages 51-58.

[49] Shen, H.W. et D.L. Kao. UFLIC: a Line Integral Convolution Algorithm for Visualizing Unsteady Flows. Dans R. Yagel et H. Hagen, Editeurs (*Proceedings de IEEE Visualization'97*, 19-24 Octobre, 1997), pages 317-322, IEEE Press.

[50] Shoup, R. Color Table Animation. Dans *IEEE Computer Graphics*, Vol 13, No 4, 1979, pages 8-13.

[51] Shroeder, W.J., C.R. Volpe et W.E. Lorensen. The Stream Polygon: A Technique for 3D Vector Field Visualization. Dans G.M. Nielson et L. Rosenblum, Editeurs (*Proceedings de Visualization'91*, 22-25 Octobre, 1991), pages 126-132, IEEE Press.

[52] Stalling, D. et H-C. Hege. Fast and Resolution Independant Line Integral Convolution. Computer Graphics Annual Conference Series (*Proceedings de SIGGRAPH'95*, 6-11 Août, 1995), pages 249-258, ACM Press.

[53] Stalling, D., M. Zöckler et H-C. Hege. Fast Display of Illuminated Field Lines. *IEEE Transactions on Visualization and Computer Graphics*, No. 2, Vol. 3, 1997, pages 118-128.

[54] Stolk, J. et J.J. van Wijk. Surface Particles for 3D Flow Visualization. Dans F.H. Post et A.J. Hin, Editeurs (*Advances in Scientific Visualization*, 1992), pages 119-130, Springer.

[55] Turk, G. et D. Banks. Image-Guided Streamline Placement. Computer Graphics Annual Conference Series (*Proceedings de SIGGRAPH'96*, 4-9 Août, 1996), pages 453-460, ACM Press.

[56] Ueng, S.K., K. Sikorski et K-L. Ma. Fast Algorithms for Visualizing Fluid Motion in Steady Flow on Unstructured Grids. *Technical Report*, Institute for Computer Applications in Science and Engineering, Number TR-95-58, Août 1995.

[57] Wegenkittl, R., E. Gröller et W. Purgathofer. Animating Flowfields: Rendering of Oriented Line Integral Convolution. *Computer Animation'97*, Juin 1997, pages 15-21, IEEE Computer Society.

[58] Wegenkittl, R. et E. Gröller. Fast Oriented Line Integral Convolution for Vector Field Visualization via the Internet. Dans R. Yagel et H. Hagen, Editeurs (*Proceedings de IEEE Visualization'97*, 19-24 Octobre, 1997), pages 309-316, IEEE Press.

[59] van Wijk, J. J. Spot Noise-Texture Synthesis for Data Visualization. Computer Graphics 25(4) (*Proccedings de SIGGRAPH'91*, 28 Juillet - 2 Août, 1991), pages 309-318, ACM Press.

[60] van Wijk, J. J. Rendering Surface Particles. Dans Arie E. Kaufman et Gregory M. Nielson, Editeurs (*Proceedings de IEEE Visualization'92*, 19-23 Octobre, 1992), pages 54-61, IEEE Press.

[61] van Wijk, J. J. Implicit Stream Surfaces. Dans G.M. Nielson et R. Dan Bergeron, Editeurs (*Proceedings de IEEE Visualization'93*, 25-29 Octobre, 1993), pages 245-252, IEEE Press.

[62] Zöckler, M., D. Stalling et H.-C. Hege. Interactive Visualization of 3D-Vector Fields Using Illuminated Stream Lines. Dans R. Yagel et G.M. Nielson, Editeurs (*Proceeding de Visualization'96*, 27Octobre - 1 Novembre, 1996), pages 181-195, IEEE Press.

Résumé

Les champs de vecteurs sont couramment utilisés en science et en ingénierie car ils permettent de coder des phénomènes physiques tels que l'écoulement des fluides, les champs électromagnétiques ou acoustiques, ou le comportement des systèmes dynamiques. Une manière de comprendre ces phénomènes est de générer des représentations visuelles des champs de vecteurs afin de révéler leur information directionnelle, la position des «points critiques» et leur évolution temporelle. Nos travaux concernent la visualisation de champs de vecteurs bidimensionnels et nous proposons de nouvelles méthodes pour la représentation de telles structures par des ensembles de streamlines, ligne tangentes aux vecteurs en tout point. En particulier nous montrons comment organiser de tels ensembles afin d'obtenir les meilleurs résultats visuels ainsi que leur utilisation pour l'animation de champs de vecteurs stationnaires ou non.

Une première étape a consisté à développer un algorithme efficace de placement de streamlines afin de couvrir uniformément le domaine a visualiser. La densité des représentations peut être facilement contrôlée par l'utilisateur en spécifiant la distance de séparation entre les streamlines. Une variante de cet algorithme permet de produire des ensembles de streamlines multi-résolution représentant l'information à différents niveaux de détails, tout en conservant une densité uniforme à chaque niveau de la hiérarchie.

Quand les vecteurs représentent des vitesses, pour un fluide par exemple, la méthode la plus naturelle pour rendre les vélocités consiste à produire une animation montrant l'écoulement de ce fluide au cours du temps. Nous avons développé une technique appelée Motion Map (carte de mouvement) qui permet de calculer en quelques secondes des animations de haute qualité visuelle à partir de n'importe quel champ de vecteurs stationnaire. Une caractéristique importante est le faible encombrement mémoire de nos représentations puisqu'une animation complète est encapsulée dans un format d'image statique.

Nous avons enfin étendu notre algorithme de placement au cas des champs de vecteurs non stationnaires. Leur évolution temporelle est visualisée par une animation ou les ensembles de streamlines successifs sont corrélés en forme et en position. L'orientation des vecteurs le long des streamlines est rendu par le déplacement de motifs corrélés.

Abstract

Vector fields are commonly used in science and engineering as they can encode physical phenomenas such as flows, electromagnetic or acoustic fields, or the behavior of dynamical systems. A means to understand these phenomenas is to create visual representations of vector fields to depict their directional information, the location of the «critical points» and their temporal evolution. Our work concerns the visualization of bidimensional vector fields and we propose new methods to represent such structures based on sets of streamlines, curves everywhere tangent to the vector field. In particular, we show how to organize these sets of streamlines to achieve better visual results and demonstrate their use to animate vector fields, whether stationary or not.

A first step consisted in the development of an efficient streamline placement algorithm that uniformly covers the domain to visualize. The density of the representation is easily controlled by the user by specifying the separating distance between streamlines. A modified algorithm can produce multiresolution sets of streamlines that represents the information at different levels of detail, while conserving a uniform density at each level of the hierarchy.

The most natural representation of velocity and orientation is based on animations. We developed a technique, called Motion Map, which computes, in a matter of seconds, high quality animations from any steady vector field. An important feature of our representation is its low memory cost; a complete animation is embedded in a static image file format.

Finaly, we have extended our placement algorithm to unsteady vector fields. Their temporal evolution is visualized through an animation where the sets of successive streamlines in time are correlated together in shape and position. The vector orientation is rendered by the displacement of colored patterns.