Titre: Title:	Sélection des points de vue pour des données vectorielles en utilisant des techniques du réseau de neurones
Auteur: Author:	Zhenyu Yang
Date:	2022
Type:	Mémoire ou thèse / Dissertation or Thesis
Référence: Citation:	Yang, Z. (2022). Sélection des points de vue pour des données vectorielles en utilisant des techniques du réseau de neurones [Master's thesis, Polytechnique Montréal]. PolyPublie. https://publications.polymtl.ca/10284/

URL de PolyPublie: PolyPublie URL:	https://publications.polymtl.ca/10284/
Directeurs de recherche: Advisors:	Benoît Ozell
Programme: Program:	Génie informatique

POLYTECHNIQUE MONTRÉAL
affiliée à l'Université de Montréal

Sélection des points de vue pour des données vectorielles en utilisant des techniques du réseau de neurones

ZHENYU YANG
Département de génie informatique et génie logiciel

Mémoire présenté en vue de l'obtention du diplôme de *Maîtrise ès sciences appliquées*
Génie informatique

Avril 2022

POLYTECHNIQUE MONTRÉAL
affiliée à l'Université de Montréal

Ce mémoire intitulé :

Sélection des points de vue pour des données vectorielles en utilisant des techniques du réseau de neurones

présenté par **Zhenyu YANG**
en vue de l'obtention du diplôme de *Maîtrise ès sciences appliquées*
a été dûment accepté par le jury d'examen constitué de :

Ettore MERLO, président
Benoît OZELL, membre et directeur de recherche
Quentin CAPPART, membre

DÉDICACE

À mes parents et à tous mes amis

REMERCIEMENTS

J'aimerais remercier spécialement mon directeur Benoît Ozell de m'avoir amené dans le domaine de recherche de l'infographie. Toutes ses aides fournies pendant mes travaux de recherche et ses suggestions sur la rédaction du mémoire me font connaître mieux la démarche de la recherche et m'incitent à m'engager plus dans cette dernière.

RÉSUMÉ

La sélection des points de vue est un domaine de recherche important qui développe les méthodologies pour générer des visualisations sur des données numériques afin que les scientifiques puissent conclure rapidement des informations utiles en partant de ces images fournies.

Dans ce mémoire, nous introduisons une nouvelle méthode qui trouve de bons points de vue pour des données vectorielles en construisant un réseau de neurones sous la forme d'un seul perceptron. Ce réseau de neurones simule le produit scalaire entre les poids du réseau et l'entrée, et est entraîné par nos nouvelles fonctions de perte qui représentent les critères sur la perpendicularité, la colinéarité et la distance représentée par le cosinus de l'angle entre deux vecteurs. Le vecteur obtenu par la normalisation des poids avec la valeur de perte minimale est la meilleure direction de vue trouvée pour visualiser le champ vectoriel. Un point de vue de cette direction est finalement construit avec la méthode de construction de point de vue proposée. Nous fournissons aussi une façon simple pour construire un chemin de caméra en utilisant des splines cubiques.

Notre méthode trouve plusieurs bons points de vue pour les ensembles de données testés en quelques secondes quand la fonction de perte de la colinéarité n'est pas activée. Si cette dernière est utilisée, notre méthode peut fournir un point de vue fiable de bonne qualité. Son temps d'exécution est beaucoup plus court que les méthodes habituelles avec l'entropie de point de vue.

ABSTRACT

Viewpoint selection is an important task in the domain of scientific visualization and can help scientists easily obtain information from the gathered data.

In this thesis, we introduce a new method that utilizes a neural network to find an ideal viewpoint for the considered data in form of vector fields. Our method constructs a single perceptron that simulates the dot product between the weights considered as the viewpoint direction and the input vector, and we train this network by using our newly defined "perpendicularity loss", "same line loss" and "cos distance loss". The normalized vector from the weights with the minimum loss value after training is a good viewpoint direction referring to the criteria represented by the activated loss functions. A final viewpoint is constructed by using the proposed viewpoint generation method. We also present a simple way to build a camera path with spline curves.

Our method finds multiple good viewpoints for the tested vector fields in a few seconds for some of our defined use cases and provides trustworthy viewpoints when the same line loss is activated. It is much faster than the previously proposed methods using viewpoint entropy.

TABLE DES MATIÈRES

LISTE DES TABLEAUX

LISTE DES FIGURES

LISTE DES SIGLES ET ABRÉVIATIONS

VE Viewpoint Entropy

LISTE DES ANNEXES

CHAPITRE 1 INTRODUCTION

De nos jours, nous générons de plus en plus de données 3D durant les activités de recherche, par exemple, les structures des molécules [2], les lignes de courant décrivant une tornade [1], et le modèle du corps humain montrant des tumeurs [3]. Elles sont souvent sous la forme d'un simple modèle 3D, d'une scène 3D, de lignes de courant ou d'un champ vectoriel. La visualisation scientifique est un outil essentiel qui aide les scientifiques à retirer de l'information des données 3D collectées. Elle se décompose en deux sous-tâches principales : la sélection des points de vue et la génération du chemin de caméra.

Le champ vectoriel est un type de données spéciales construites par des vecteurs qui décrivent, par exemple, les directions du mouvement des fluides ou les phénomènes naturels concernant le climat. Les publications récentes sur la sélection des points de vue se portent souvent sur les lignes de courant et les modèles 3D mais rarement sur les données vectorielles [1–7]. De plus, la plupart des méthodes développées se basent sur l'entropie de point de vue, qui est une mesure générale pour choisir un point de vue, et qui possède en même temps une forte dépendance sur la méthode de rendu. Une petite modification dans le pipeline de rendu peut provoquer une différence sur le point de vue sélectionné.

Notre but est de concevoir une nouvelle méthode de la sélection des points de vue qui ne possède pas les défauts de celles précédemment proposées en générant automatiquement de bons points de vues pour observer des données vectorielles avec juste les informations géométriques de ces dernières et de proposer une méthode simple utilisant les courbes polynomiales afin de construire un chemin de caméra avec des points de vue générés pour réaliser une navigation entre eux. Nous réalisons la méthode désirée sur la sélection de point de vue en entraînant un seul perceptron avec les données vectorielles et puis en retirant le poids en tant que la direction de vue pour construire un point de vue final. La génération du chemin de caméra est faite en connectant les multiples points de vue par des splines cubiques.

CHAPITRE 2 REVUE DE LITTÉRATURE

Nous vivons dans un monde où les données jouent un rôle crucial pour son fonctionnement. De plus en plus de données créées dans l'industrie ou dans les activités de recherche possèdent des formes complexes et difficiles à percevoir. Une visualisation adéquate des données générées est essentielle pour l'acquisition des informations importantes pour le travail. Nous nous intéressons à des algorithmes qui génèrent ce genre de visualisation dans le domaine de l'infographie.

2.1 La visualisation des données

La visualisation des données est l'ensemble des méthodes qui transforment des données collectées vers une représentation graphique afin de faciliter la compréhension et l'analyse des informations. Elle consiste à créer des figures ou des animations qui illustrent correctement et efficacement les informations importantes. Nous pouvons généralement la séparer en deux genres : la visualisation des données en 2D et celle en 3D. La visualisation des données en 2D construit normalement des diagrammes (Diagramme en bâtons, Diagramme circulaire, Nuage de points, Diagramme à bulles, etc.) et celle de 3D montre plutôt des vues sur les données considérées ou des animations qui aident à faire un parcours au sein des données (Des exemples sont montrés dans la Figure 2.1).

2.2 La visualisation scientifique

Ce mémoire se focalise dans un sous-domaine de la visualisation de données qui est la visualisation scientifique, cette visualisation est liée fortement aux données 3D générées durant des recherches scientifiques ou par des capteurs utilisés dans l'industrie. Ces données sont souvent immense et possèdent des natures géométriques complexes, et une perception intuitive ne permet pas de reconnaître des informations efficaces et utiles. Des outils d'exploration comme ParaView [1] ou VU [2] fournissent la possibilité d'explorer ces données attentivement et de connaître leur nature étape par étape. Cependant, les méthodes qui trouvent automatiquement des vues appropriées pour acquérir rapidement des informations sont moins développées.

1. Ahrens, James, Geveci, Berk, Law, Charles, ParaView : An End-User Tool for Large Data Visualization, Visualization Handbook, Elsevier, 2005, ISBN-13 : 978-0123875822
2. `https://invisu.ca/`

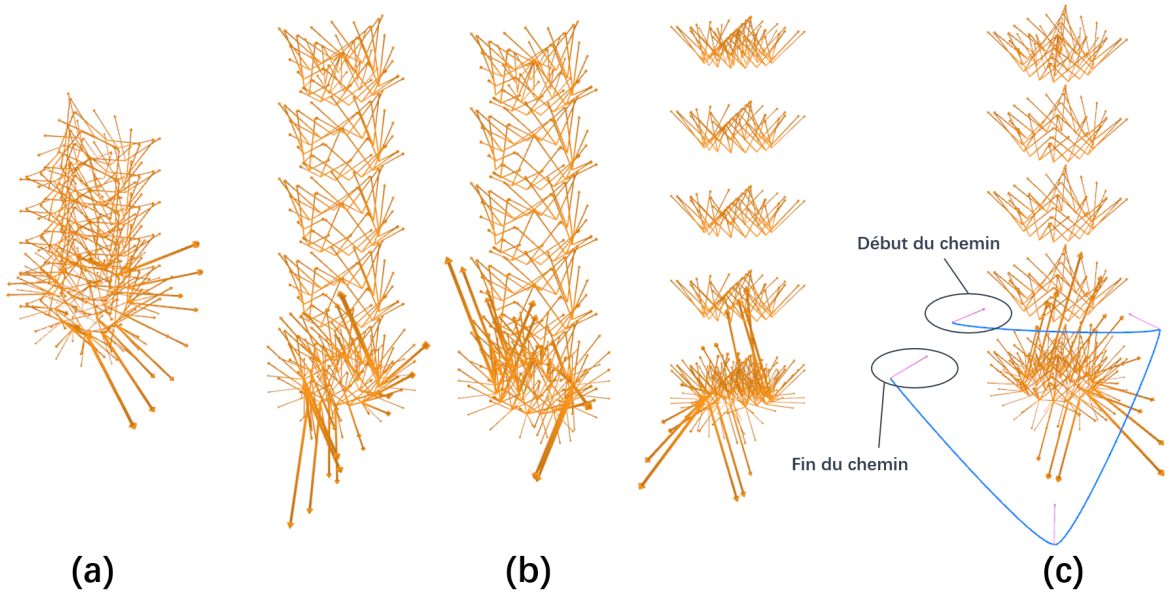


Figure 2.1 Exemple des visualisations en 3D. (a) montre les données 3D à visualiser qui forment un champ vectoriel. (b) montre des points de vue possibles pour ces données. (c) est un chemin de caméra possible pour générer une animation qui réalise un parcours.

Les données 3D à visualiser pour la visualisation scientifique sont principalement de ces types :

- Scène 3D : Une scène composée par des objets intéressants et des obstacles (constructions, arbres, transports et meubles, etc.).
- Modèle 3D : Un modèle 3D possédant une forme géométrique irrégulière à observer.
- Lignes de courant : Des lignes de flux qui indique le mouvement des fluides (fumée, air, lignes magnétiques, etc.)
- Champ de vecteurs (données vectorielles) : Un champ de vecteurs qui indique des directions de mouvement.

2.3 La sélection des points de vue

Dans un premier temps, nous réviserons des méthodes existantes pour sélectionner des bons points de vue des données. Il existe quatre grands groupes d'algorithmes pour cette tâche.

2.3.1 Première famille : Entropie de point de vue

Une grande majorité des méthodes sont basées sur l'entropie de point de vue proposée par Vázquez et al. [2]. Cette entropie est inspirée l'entropie de Shannon qui est une mesure bien commune pour quantifier l'information. Elle est représentée sous la forme de cette équation :

$$H(X) = -\sum_{i=1}^{n} p_i log(p_i) \tag{2.1}$$

X est une variable aléatoire et p_i est la probabilité dans le cas où X égale à une valeur possible a_i pour X. Le terme $-log(p_i)$ est l'information portée par cette valeur, car il est inversement proportionnel à p_i, c'est-à-dire il possède une grande valeur quand a_i est rare et une petite valeur quand a_i est commune. Ainsi, cette équation calcule une moyenne pondérée de l'information pour les valeurs possibles de X qui donne une perception sur l'information obtenue quand nous lançons une expérience.

En partant de cette entropie, Vázquez et al. [2] proposent une façon pour mesurer l'information fournie par un point de vue. Ils considèrent une sphère englobante unitaire autour du point de vue désiré, et tous les objets dans la scène sont projetés sur la surface de cette sphère. Leurs surfaces projetées sont notées par A_i pour chaque surface i. Et $\frac{A_i}{4\pi}$ symbolise la probabilité pour voir une surface sur la sphère englobante qui est le champ de vision total pour ce point de vue. En injectant cette probabilité dans l'entropie de Shannon, ils obtiennent l'entropie de point de vue :

$$I(S,p) = -\sum_{i=1}^{N_f} \frac{A_i}{4\pi} log(\frac{A_i}{4\pi}) \tag{2.2}$$

N_f est le nombre total des surfaces des objets dans la scène. Cette entropie mesure l'information moyenne acquise en regardant la scène avec une direction à partir d'un point de vue. Avec cette propriété, elle peut devenir une mesure pour évaluer la qualité d'un point de vue.

Cependant, cette formule reste toujours abstraite. Afin de fournir une façon simple pour calculer l'entropie de point de vue en prenant une image affichée après une projection orthogonale, Vázquez et al. [2] proposent une autre forme de l'entropie de point de vue qui est discrétisée jusqu'au niveau du pixel :

$$I_o(S,p) = -\sum_{i=1}^{N_f} \frac{N_{pixi}}{N_{pixf}} log(\frac{N_{pixi}}{N_{pixf}}) \tag{2.3}$$

N_{pixi} est le nombre de pixels visualisés d'une surface i sur l'image projetée et N_{pixi} est le nombre de pixels total de l'image. Cette forme discrétisée de l'entropie de point de vue mesure l'information moyenne possédée par un pixel de l'image projetée qui peut aussi être

considérée comme l'information de l'image projetée pour un point de vue. Cette nouvelle forme jusqu'au niveau de pixel simplifie beaucoup le calcul en tant qu'une bonne mesure de qualité de point de vue.

Vázquez et al. [7] utilisent une méthode similaire basée sur l'entropie de point de vue comme celle dans [2], mais le domaine d'application est différent. Cela montre la variété d'utilisations de cette mesure.

En 2012, Monclús et al. [4] ont appliqué une transformée en ondelettes discrète sur l'image obtenue avec un nombre de niveaux fixé et ont construit encore une version modifiée de l'entropie de point de vue étant moins sensible aux bruits dans l'image (On considère toujours l'image en RGB) :

$$H_w(X) = -\sum_{l=1}^{L}\sum_{k=0}^{N_l} p_{w_{l,k}} log(p_{w_{l,k}}) \tag{2.4}$$

$p_{w_{l,k}}$ est la valeur relative entre le coefficient de chaîne l et la valeur k obtenue dans la chaîne l.

L'utilisation de ces entropies est basée sur un ensemble de points de vue générés sur une sphère englobante pour le modèle 3D à visualiser, et la valeur d'entropie est calculée pour chaque point de vue sur la sphère. Le meilleur point de vue sera le point de vue avec une entropie maximale. Évidemment, plusieurs bons points de vue avec des entropies secondaires peuvent être sélectionnés pour la génération de chemin de caméra ou l'observation multiple du modèle.

L'application de ces entropies reste toujours une sélection des points de vue selon un seul critère, qui ne porte pas forcément un bon résultat et qui est un peu restreinte. Afin de résoudre des problèmes complexes pour la planification des interventions médicales, Muehler et al. [3] proposent une nouvelle méthode de sélection avec multi-critères qui élargit l'utilisation de l'entropie de point de vue.

Dans cette méthode, une matrice de taille $(n+1) \times n$ qui enregistre les nombres des pixels occlus entre les objets, les nombres totaux de pixels visibles pour chaque objet et les nombres de pixels non occlus pour chaque objet est calculée dans un premier temps, elle sera utilisée pour évaluer certaines des critères considérés. Ces derniers sont :

- Entropie d'objet : L'entropie de point de vue calculée avec des surfaces projetées des objets sans considération des occlusions.
- Nombre d'objets qui occluent les autres : Pour ce critère, la valeur doit être la plus petite possible pour un bon point de vue.
- Importance des objets qui occluent les autres : La somme des ratios entre les surfaces occluses et les importances des objets qui les occluent.
- Taille des surfaces non occluses.

— Région préférée : Ce critère indique la région ou l'angle préféré pour observer le modèle.
— Distance au point de vue courant : Ce critère est dans le but de stabiliser la visualisation et d'éviter des sauts soudains vers une région éloignée.
— Stabilité des points de vue : Un tableau qui contient des valeurs représentant la distance aux bordures de l'espace « monde » pour un point de vue.

En prenant ces critères, Muehler et al. [3] calculent une somme pondérée de ces derniers :

$$S(\alpha, \beta) = \sum_{i=1}^{n} w_i p_i(\alpha, \beta) \tag{2.5}$$

α et β sont les coordonnées sphériques pour un point de vue, $p_i(\alpha, \beta)$ est la valeur du critère i pour ce point de vue et w_i est le poids constant à multiplier pour ce critère. Cette somme pondérée devient leur mesure finale pour sélectionner des points de vue. Les bons points de vue seront sélectionnés en considérant le tableau formé par les valeurs calculées de cette mesure pour tous les points de vue générés.

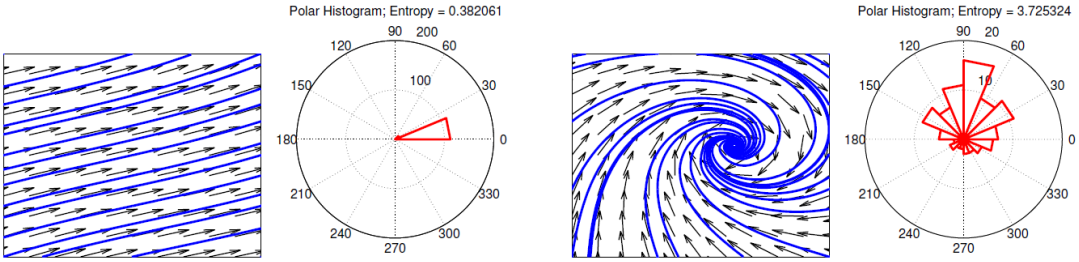
Cette méthode est très spécifique pour leur cas de recherche, mais reste inspirante. Elle fournit l'idée de multi-critères appropriés qui améliore potentiellement l'effet final de sélection. Cependant, plusieurs critères dans leur méthode demandent des interventions des experts, par exemple, la détermination des importances des objets dans la scène en tant que l'objet qui occlut les autres. Ce processus n'est pas assez « automatique ».

2.3.2 Deuxième famille : Utilisation variante de l'entropie de Shannon

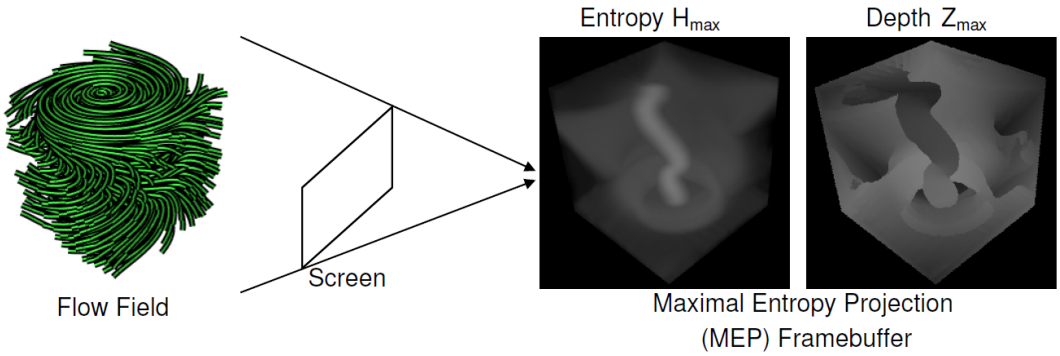
En passant notre focus du modèle 3D vers les lignes de courant, une différente façon d'utilisation de l'entropie de Shannon apparaît.

Lee et al. [1] utilisent l'entropie de Shannon pour évaluer la complexité des flux autour d'un point du champ vectoriel.

En appliquant une voxelisation du modèle des lignes de courant considéré, la scène devient un ensemble de voxels. Pour chaque voxel, ils construisent un histogramme polaire qui compte le nombre de vecteurs au voisinage dans chaque intervalle de direction. En prenant les probabilités obtenues à partir de cet histogramme pour chaque intervalle de direction, l'entropie représentant la complexité des flux de ce voxel est ensuite calculée en utilisant l'entropie de Shannon. Les valeurs d'entropie pour tous les voxels sont enregistrés dans une carte de valeurs. Après un lancer de rayons à partir de chaque pixel de la fenêtre pour un point de vue considéré, un tampon des valeurs des entropies maximales pour chaque pixel en suivant la direction du rayon correspondant est construit, il est nommé H_{max}. Ce processus est illustré dans la Figure 2.2.



(a) Histogramme polaire autour d'un voxel



(b) Lancer de rayons pour obtenir H_{max}

Figure 2.2 Calcul de la complexité des flux (images provenant de [1])

Ils génèrent des points de vue en tant que candidats sur une sphère qui entoure les lignes de courant. Pour chaque point de vue, ils appliquent le processus de lancer de rayons afin d'obtenir le tampon H_{max} correspondant. La somme des valeurs du tampon H_{max} pour chaque point de vue est ensuite calculée et elle est considérée comme la mesure finale pour sélectionner des bons points de vue.

Dans cet article, les auteurs proposent aussi une méthode pour sélectionner des lignes de courant importantes qui représentent la squelette du modèle original en utilisant des densités de lignes calculées. Cette sélection améliore la visualisation des lignes de courant en enlevant des parties qui provoquent des occlusions et qui dérangent la compréhension des informations.

Cette variante de l'entropie de Shannon fournit une nouvelle idée pour l'application de cette entropie et elle peut être utilisée pour mesurer la complexité des parties d'un champ de vecteurs afin de trouver des régions intéressantes.

2.3.3 Troisième famille : Modèle de probabilité

Quand nous sortons des méthodes formalisées par les entropies, nous découvrons une méthodologie totalement différente.

Tao et al. [5] pensent la sélection des points de vue comme une question de probabilité. Les auteurs ont construit deux chaînes de probabilité :

— $V \rightarrow S$: On considère des points de vue aux lignes de courant. La probabilité de transition pour cette chaîne est $p(s \mid v)$, elle symbolise la probabilité de « voir » la ligne de courant s en se situant au point de vue v.

— $S \rightarrow V$: On considère des lignes de courant aux points de vue. La probabilité de transition pour cette chaîne est $p(v \mid s)$, elle symbolise la probabilité de choisir le point de vue v considérant la ligne de courant s.

Afin de modéliser ces deux chaînes de probabilité, ils ont déterminé en premier $p(s \mid v)$. Pour ce faire, ils ont découpé la ligne de courant s et sa projection s_v en des ensembles de points. En faisant des histogrammes individuels et conjoints sur les vecteurs interpolés de ces points de s et s_v, ces probabilités sont calculées :

— Probabilité pour « voir » un point de s ou s_v. Elle est la probabilité du vecteur interpolé partant de ce point calculée avec les nombres comptés dans les histogrammes individuels.

— Probabilité pour « voir » un groupe de deux points se situant séparément sur s et s_v. Elle est la probabilité du groupe des vecteurs interpolés partant de ces deux points calculée avec les nombres comptés dans les histogrammes conjoints.

Avec ces probabilités marginales et conjointes, l'information de la ligne de courant s est calculée. En faisant la division entre l'information de la ligne de courant s et la somme des informations de toutes les lignes de courant dans la scène avec les coefficients du principe de visualisation de 45 degrés, ils ont défini $p(s \mid v)$.

À partir de cette probabilité de transition, la probabilité pour sélectionner un point de vue v est définie :

$$p(v) = \frac{p(S \mid v)}{p(S \mid V)} \; avec \; p(S \mid v) = \sum_{\forall s \in S} p(s \mid v), p(S \mid V) = \sum_{\forall v \in V} p(S \mid v) \tag{2.6}$$

Cette probabilité est une normalisation de la probabilité pour « voir » toutes les lignes de courant sur le point de vue v par rapport à la somme de ce type de probabilité pour tous les points de vue.

$p(v)$ devient déjà une bonne mesure pour sélectionner un point de vue, car elle est la probabilité pour prendre ce point de vue considérant toute la scène des lignes de courant. Cependant,

les auteurs ont proposé une autre mesure qu'ils nomment l'information mutuelle de point de vue :

$$I(v;S) = \sum_{\forall s \in S} p(s \mid v) log \frac{p(s \mid v)}{p(s)} \tag{2.7}$$

Cette nouvelle mesure représente le niveau de dépendance entre le point de vue considéré et l'ensemble des lignes de courant. Elle est aussi une bonne mesure qui exprime l'intention pour utiliser v pour visualiser la scène.

L'autre chaîne $p(v \mid s)$ est construite en utilisant le théorème de Bayes. Avec cette probabilité de transition, des mesures comme $p(s)$ et $I(s;V)$ pour sélectionner des lignes sont définies. Elles aident à améliorer l'image du point de vue en montrant seulement les lignes de courant importantes ou en tant que squelette.

Cette méthodologie souligne l'importance du lien entre l'objet lui-même et sa projection et la possibilité de l'application des modèles de probabilité dans la sélection des points de vue. Ce point est inspirant et à considérer pour les futures recherches.

2.3.4 Quatrième famille : Apprentissage profond

Durant les dernières années, l'apprentissage profond devient très populaire en tant que l'outil pour automatiser des processus. Il peut être aussi appliqué pour la sélection du point de vue.

Nous révisons en premier des concepts généraux pour l'apprentissage profond.

Réseau de neurones

Le réseau de neurones est un réseau formulé par des « neurones » mathématiques qui simulent un calcul spécifié et qui représente une fonction $f(\mathbf{x})$ pour un vecteur d'entrée $\mathbf{x}$.

Perceptron

Le « neurone » mathématique dans le réseau de neurones est aussi appelé perceptron. Le perceptron possède la structure illustrée dans la Figure 2.3.

Les composants d'un perceptron sont :

- Vecteur d'entrée $\mathbf{x}$
- Vecteur $\mathbf{w}$ qui contient les valeurs à multiplier sur chaque attribut de $\mathbf{x}$, il est aussi appelé le poids du perceptron.
- Vecteur $\mathbf{b}$ qui contient les valeurs à ajouter sur le produit $\mathbf{w} \cdot \mathbf{x}$, il est aussi appelé le biais du perceptron.

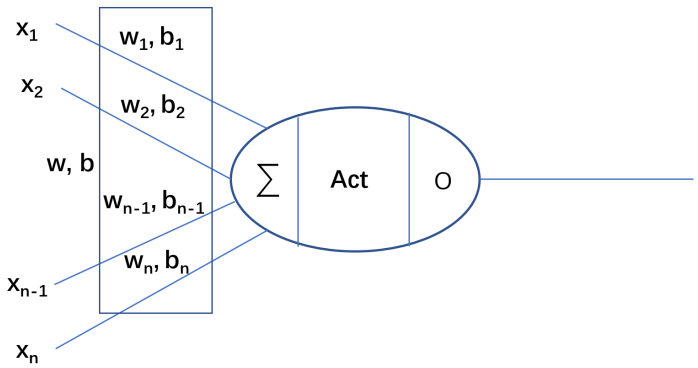


Figure 2.3 Perceptron du réseau de neurones

— Fonction d'activation *Act* qui sert à modifier la valeur après la sommation. Elle fournit la non-linéarité du perceptron.
— Valeur de sortie o.

Le perceptron réalise le produit scalaire $\mathbf{x} \cdot \mathbf{w}$ et y ajoute le biais $\mathbf{b}$, et puis active le résultat du produit par la fonction d'activation, la valeur activée devient la valeur de sortie. En équation, ce calcul s'écrit :

$$o = Act(\mathbf{w} \cdot \mathbf{x} + \mathbf{b}) \tag{2.8}$$

Perceptron multicouche

Un réseau de neurones contenant des couches multiples qui sont connectées entre eux par les entrées et les sorties s'appelle un perceptron multicouche [8, Chapitre 6]. Il a souvent la forme illustrée dans la Figure 2.4.

Réseau de neurones convolutif

Ce type de réseau [8, Chapitre 9] contient des couches non simplement construites par des perceptrons, mais aussi des couches qui appliquent des calculs de convolution. Le poids de ce type de couche est un noyau commun pour appliquer une convolution discrète sur une image. La sortie de ces couches est fréquemment une image contenant des propriétés de l'image originale, par exemple, les contours des objets. Cette sortie est souvent connectée sur un perceptron multicouche afin de réaliser une tâche de classification.

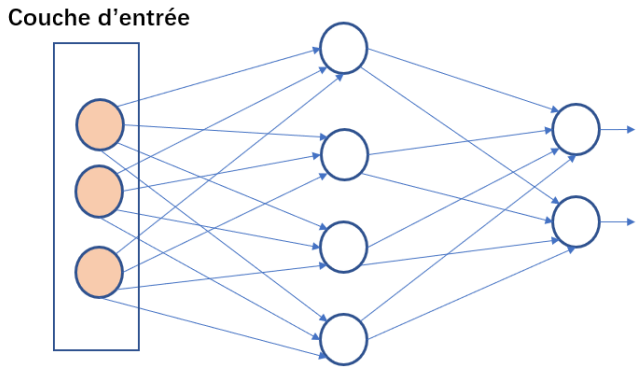


Figure 2.4 Structure du perceptron multicouche

Fonction de perte

Pour entraîner le réseau de neurones, nous devons définir une fonction de perte. Nous considérons ici le cas de l'apprentissage supervisé. Le but est de faire apprendre le réseau de neurones à produire des valeurs de sortie qui sont similaires ou identiques à celles que nous indiquons pour chaque entrée $\mathbf{x}$. Les valeurs que nous indiquons pour les entrées sont appelées « cibles ». La fonction de perte définit la distance entre les valeurs de sorties et les cibles pour toutes les entrées. Les fonctions de perte les plus utilisées sont :

— La fonction de perte quadratique (en anglais « Mean Square Error ») :

$$L_{quad} = \frac{1}{2N} \sum_{\forall \mathbf{x} \in X} (\mathbf{cible} - f_{reseau}(\mathbf{x}))^2 \tag{2.9}$$

où N est le nombre total des entrées et X est l'ensemble d'entraînement.

— L'entropie croisée :

$$L_{EC} = -\frac{1}{N} \sum_{\forall \mathbf{x} \in X} \sum_{c=1}^{M} y_{o,c}(\mathbf{x}) log p_{o,c}(\mathbf{x}) \tag{2.10}$$

où N est le nombre total des entrées, X est l'ensemble d'entraînement, M est le nombre total des classes, $y_{o,c}(\mathbf{x})$ est l'indicateur binaire (valeur 0 ou 1) pour montrer la classe correcte de $\mathbf{x}$ (Le vecteur formé par les valeurs de $y_{o,c}(\mathbf{x})$ pour toutes les classes est la cible) et $p_{o,c}(\mathbf{x})$ est la probabilité prédite par le réseau pour la classe c considérant $\mathbf{x}$.

Méthode d'entraînement

Pour une étape d'entraînement, nous faisons passer toutes les entrées par le réseau afin d'obtenir les valeurs de sorties correspondantes. La valeur de perte actuelle est ensuite calculée avec ces sorties et les cibles des entrées. Avec cette valeur de perte, le processus de la propagation en arrière (en anglais « back propagation ») est appliquée en vue d'obtenir les gradients pour les poids de toutes les couches. Les poids de toutes les couches sont renouvelés selon l'optimiseur choisi en utilisant les gradients trouvés et le taux d'apprentissage sélectionné.

Après plusieurs époques, le réseau possédera un poids total du réseau qui minimise localement ou globalement la fonction de perte, c'est-à-dire qu'il donne (ou prédit) une valeur de sortie similaire à la cible pour chaque entrée.

Optimiseur

Nous listons ici les méthodes d'optimisation fréquemment utilisées pour optimiser un réseau de neurones :

— Algorithme du gradient stochastique (en anglais, *Stochastic gradient descent(SGD)*) [8, Chapitre 8] : La façon la plus simple pour faire apprendre le réseau, les poids sont renouvelés par cette équation :

$$\mathbf{w} = \mathbf{w} - lr \times \mathbf{gradient} \tag{2.11}$$

— SGD-momentum [8, Chapitre 8] : Cet optimiseur renouvelle les poids du réseau selon un gradient interpolé entre la somme des anciens gradients et le gradient actuel. Cette modification ajuste le gradient vers l'ancienne direction mémorisée tendant vers le minimum local ou global afin que le processus d'optimisation soit plus rapide.

— Adagrad [9] : Cet optimiseur renouvelle les poids du réseau selon un gradient ajusté par la racine des moments quadratiques accumulés jusqu'à présent. Cet ajustement fournit un mécanisme d'incitation qui augmente le pas d'avancement au début de l'entraînement et qui diminue le pas d'avancement quand on est proche du minimum local ou global.

— Adam [10] : Cet optimiseur renouvelle les poids du réseau selon un gradient ajusté calculé par la division entre la somme interpolée du premier moment et la racine de la somme interpolée du moment quadratique accumulées au fur et à mesure. Il donne l'effet de SGD-momentum et Adagrad.

Utilisation du réseau de neurones dans la sélection des points de vue

Avec ces techniques du réseau de neurones, Schelling et al. [6] développent une nouvelle méthode qui génère des points de vue similaire à ceux choisis par les entropies ou les autres mesures en utilisant leurs réseaux de neurones construits.

Les auteurs construisent un réseau séparé en deux parties :

- Réseau de neurones convolutif de Monte-Carlo : Ce réseau réalise plusieurs opérations de convolution de différents rayons pour extraire les informations géométriques du modèle 3D en entrée dans un vecteur de taille 2048.
- Plusieurs perceptrons multicouches liés sur la sortie du réseau convolutif : Chacun des perceptrons multicouches réalise une prédiction d'un point de vue similaire à celui obtenu par une mesure de sélection de points de vue considérée.

Les mesures de sélection de points de vue considérées sont :

- Entropie de point de vue
- Ratio de visibilité : la somme des valeurs relatives entre la surface d'un polygone et la superficie totale du modèle pour tous les polygones visibles.
- Divergence de Kullback-Leibler de point de vue : Une version modifiée de l'entropie de point de vue qui considère le ratio entre la superficie originale de polygone et la superficie totale.
- Information mutuelle de point de vue : Similaire à la mesure dans [5], les lignes de courants sont simplement remplacées par les polygones du modèle.

L'entraînement du réseau de neurones proposé se sépare en deux étapes :

- Entraînement avec les cibles multiples : Plusieurs cibles sont générées pour chaque perceptron multiple selon la mesure correspondante avec une valeur critique définie. La fonction de perte pour chaque perceptron multiple est définie comme la valeur minimale des distances cosinus entre le point de vue prédit et les cibles générées. Le réseau est ensuite entraîné en utilisant ces fonctions de perte pendant des époques.
- Entraînement avec la cible gaussienne : Une région gaussienne à proximité de la prédiction à la fin de l'entraînement avec les cibles multiples est construite pour chaque perceptron multiple. Le point de vue avec une meilleure valeur selon la mesure correspondante est choisi comme la nouvelle cible. La distance cosinus entre le point de vue prédit et cette cible devient la nouvelle fonction de perte pour chaque perceptron multiple. Le réseau est ensuite entraîné avec cette fonction de perte jusqu'à la fin.

Le réseau finalement entraîné est capable de produire rapidement des points de vue similaires à ceux choisis selon les mesures considérées. Il peut être utilisé pour sélectionner des points de vue de bonne qualité sur des modèles inconnus.

Cette nouvelle méthode suggère l'idée de la simulation du processus de sélection de points de vue selon une mesure directement par le réseau de neurones. Les recherches futures peuvent profiter de ce point pour automatiser le processus de la sélection de points de vue selon de nouvelles mesures. Cette méthode nous incite à utiliser des techniques du réseau de neurones dans notre travail. Cependant, au lieu de construire un réseau de neurones complexe pour simuler la tâche de sélection, nous entraînerons plutôt un seul perceptron afin de trouver une direction de vue appropriée pour visualiser le champ vectoriel. Ce processus d'entraînement peut aussi être considéré comme une régression logistique avec des fonctions de perte conçues spécifiquement et représentant des critères de sélection de points de vue.

2.4 La génération du chemin de caméra

Si plusieurs bons points de vue sont sélectionnés, une animation d'un chemin de caméra construit sur ces derniers peut aider les scientifiques à mieux regarder dynamiquement les points de vue et à profiter de la transition entre les points de vue pour observer le modèle. Ainsi, la génération du chemin de caméra devient un thème important dans la visualisation scientifique.

Les méthodes précédemment développées pour cette tâche se découpent en trois groupes.

2.4.1 Premier groupe : Méthode basée sur A*

A* est un algorithme très utilisé pour trouver rapidement un chemin avec un coût minimal dans un graphe. Si l'espace libre d'une scène 3D peut être considéré comme un graphe, l'application de cet algorithme trouve un bon chemin de caméra selon la fonction heuristique pour parcourir la scène.

Hsu et al. [11] ont développé une méthode multicritère basée sur A* qui trouve un chemin de caméra dans une scène 3D.

La méthode se découpe en trois étapes :

- Génération du graphe dans l'espace libre de la scène.
- Construction du chemin de caméra initial avec A*.
- Raffinement du chemin construit selon les forces définies.

La première étape est faite en appliquant un algorithme pour générer une arbre de sphère. Cet algorithme répète cette opération : « Prendre un point dans l'arbre avec la valeur minimale de l'axe médian (originalement il existe juste une racine avec la valeur minimale globale de l'axe médian), trouver la sphère maximale qui comble l'espace autour de ce point, choisir des points sur la sphère selon leur valeur de l'axe médian et ajouter ces points dans l'arbre en

les connectant sur le point initial ». À la fin de cet algorithme, un arbre de sphère en tant que squelette de l'espace libre est construit. Des arêtes utiles sont ajoutées sur l'arbre pour former le graphe final.

La deuxième étape est réalisée par l'algorithme A* sur le graphe généré. La fonction heuristique utilisée est souvent la distance euclidienne à la destination, mais elle peut être modifiée en fonction qui évalue l'opacité ou l'occlusion.

La troisième étape est faite en déformant le chemin construit selon les forces définies dans l'espace. Les forces sont :

- Force de ressort : le chemin est découpé en des morceaux de ressort, chaque ressort tire les points de coupe à l'extrémité afin que le chemin soit droit.
- Force d'opacité : Des zones opaques dans l'espace appliquent des forces sur le chemin de caméra pour qu'il soit éloigné de ces dernières.
- Force d'occlusion : Une force calculée selon les informations d'occlusion autour du point de vue courant, cette force éloigne la caméra des surfaces convexes pour éviter des occlusions.
- Force de visibilité : Une force calculée selon la visibilité du prochain point d'intérêt, elle attire la caméra vers le point d'intérêt ciblé.
- Pénalité verticale : Cette force diminue des mouvements verticaux.
- Répulsion du focus : Cette force repousse la caméra plus loin quand il s'éloigne d'un point d'intérêt et l'attire quand il s'approche du prochain point d'intérêt.

Toutes ces forces raffinent le chemin vers une courbe raisonnable pour observer les points d'intérêt.

L'orientation de la caméra est souvent la tangente du chemin de caméra, mais quand la caméra arrive à voir le prochain point d'intérêt, elle regarde vers ce point. L'orientation est un peu spéciale au départ, elle est la direction interpolée selon la direction vers la source et la direction vers le premier point d'intérêt.

Benyoub [12] propose une autre façon pour utiliser l'algorithme A*. L'auteur applique cet algorithme sans régression, c'est-à-dire, il applique un choix selon la fonction heuristique parmi les huit points voisins, une fois ce choix est réalisé, il part de ce nouveau point pour sélectionner le prochain point du chemin de caméra et ne considère plus les autres points antérieurs. Leur fonction heuristique considère plusieurs critères, y inclut l'occlusion. Le chemin de caméra final est construit en connectant tous les points ou les nœuds du chemin avec des B-Splines.

Toutes ces méthodes basées sur A* nous inspirent sur l'utilisation des techniques du graphe dans la génération du chemin de caméra et la considération des critères comme l'occlusion

dans cette dernière.

2.4.2 Deuxième groupe : Forces d'attraction et de répulsion

Des méthodes purement basées sur des concepts physiques existent, par exemple, celle dans [13].

Cette méthode propose une découpe de la scène 3D à visualiser en plusieurs cubes de même taille, que les auteurs nomment « voxelisation ». Après cette voxelisation spéciale, les valeurs d'intérêt des cubes sont calculées selon les objets intéressants contenus dans leur espace. Cette valeur est utilisée pour définir la force d'attraction appliquée par le cube. Chaque cube lui-même est aussi considéré comme un obstacle, il applique une force de répulsion sur la caméra dans la scène. Finalement, la force totale calculée avec la sommation de toutes ces forces est appliquée sur la caméra pour guider son mouvement dans la scène, ce guide génère enfin un chemin de caméra.

Ce type de méthode classique selon des lois de physique donne une idée naturelle pour produire un chemin de caméra. L'idée des forces est très intéressante en tant que facteur à profiter dans le processus de la génération du chemin, elle est probablement l'origine de la partie de raffinement dans [11]. Mais elle reste un peu restreint sur son extensibilité.

2.4.3 Troisième groupe : Courbes polynomiales

Des courbes polynomiales comme les courbes de Bézier, les splines cubiques et les B-splines sont les façons les plus simples pour créer un chemin de caméra entre les points de vue. Les exemples sont :

- Amamra et al. [14] proposent une méthode avec les courbes polynomiales. Ils génèrent un chemin de caméra qui connecte les points de vue avec les courbes des hodographes de Pythagore. Ce chemin est ensuite ajusté en calculant l'intersection avec les volumes englobants des objets dans la scène et puis en éloignant le chemin de ces volumes sur le plan du chemin selon les lignes étendues des intersections les plus proches.
- Benyoub [12] utilise aussi la B-spline pour lier les nœuds pour construire le chemin final.

La génération de chemins avec les courbes polynomiales est toujours une méthodologie principale à considérer étant donné son efficacité et sa simplicité.

2.5 Non résolu ou rarement utilisé

Les méthodes mentionnées pour la sélection des points de vue portent majoritairement sur des modèles 3D ou des lignes de courant. Des recherches sur les données vectorielles sont vraiment rares. De plus, la plupart des recherches se concentrent toujours sur les méthodes de la série des entropies qui sont des mesures dépendant fortement de la méthode de rendu. Un petit changement du rendu influence potentiellement le résultat final.

L'application du réseau de neurones dans la visualisation scientifique peut avancer fortement la vitesse de sélection ou même créer de nouvelles méthodes pour choisir des points de vue. Cependant, il existe peu d'articles qui utilisent cet outil intéressant.

2.6 Objectif général

Suggérer une méthodologie pour visualiser les données vectorielles, qui est indépendante de la méthode de rendu et qui utilise des techniques du réseau de neurones.

2.6.1 Objectifs spécifiques

- O_1 : Définir les critères principaux d'un bon point de vue pour des données vectorielles.
- O_2 : Construire un réseau de neurones qui accepte les vecteurs en tant que l'entrée.
- O_3 : Établir des fonctions de perte principales pour entraîner le réseau afin de trouver une direction de vue qui satisfait les critères dans O_1.
- O_4 : Proposer et implémenter une méthode de construction d'un point de vue pour la caméra à partir de la direction trouvée.
- O_5 : Proposer et implémenter une méthode de génération du chemin de caméra selon des points de vue construits.
- O_6 : Valider la méthode sur les ensembles de données.

CHAPITRE 3 DÉMARCHE DE L'ENSEMBLE DU TRAVAIL ET ORGANISATION GÉNÉRALE DE L'ARTICLE

3.1 Démarche de l'ensemble du travail de recherche

Cette recherche a été réalisée par :

- Zhenyu Yang : Étudiant en maîtrise au département de génie informatique et génie logiciel à Polytechnique Montréal.
- Benoît Ozell : Professeur au département de génie informatique et génie logiciel à Polytechnique Montréal.

Zhenyu Yang a effectué le design d'une nouvelle méthodologie qui sélectionnait des points de vue pour le champ vectoriel. Il a aussi créé une façon simple pour construire un chemin de caméra en se basant sur des points de vue multiples sélectionnés. Tous ces travaux ont été réalisés avec les points importants retenus pendant la revue de littérature et les suggestions fournies par Benoît Ozell.

Les méthodes proposées ont été testées par Zhenyu Yang avec les ensembles de données fournies par Benoît Ozell pour obtenir les résultats et analyser ses effets.

3.2 Organisation générale de l'article

3.2.1 Titre

Viewpoint selection of vector fields by using neural network techniques

3.2.2 Publication

Cet article a été soumis à la conférence IEEE VIS 2022 pour paraitre ultérieurement dans la revue *IEEE Transactions on Visualization and Computer Graphics (TVCG)*.

3.2.3 Résumé

L'article commence par une introduction sur les éléments importants de la recherche sur la visualisation scientifique, les défauts des méthodes antérieures pour la tâche de sélection de points de vue et une synthèse de nos nouvelles méthodes proposées sans ces inconvénients.

Ensuite, il présente des méthodes précédemment construites pour réaliser les tâches de la sélection des points de vue et de la génération du chemin de caméra.

Après une section qui présente les travaux antérieurs, il passe dans le corps du contenu qui est la méthodologie. La section « méthodologie » explique d'abord le principe de la perpendicularité qui est un critère défini pour choisir un bon point de vue d'un champ vectoriel (O_1). Ensuite, elle présente la structure du réseau de neurones utilisé pour trouver la bonne direction de vue (O_2). Les fonctions de perte proposées pour entraîner ce réseau sont expliquées dans la suite (O_3). La méthode d'entraînement et la méthode de construction d'un point de vue à partir de la direction de vue trouvée sont aussi explicitées dans cette section (O_4). Cette section finit par l'explication de la méthode proposée pour générer un chemin de caméra avec des splines cubiques (O_5).

Une section qui porte sur les résultats obtenus avec des tests sur les ensembles de données utilisés et une discussion de ces résultats est ensuite présentée afin de montrer les points de vue générés de bonne qualité et les facteurs d'accélération de la méthodologie mentionnée (O_6).

À la fin, cet article offre des améliorations futures potentielles et donne une brève conclusion des contributions réalisées.

CHAPITRE 4 ARTICLE 1 : VIEWPOINT SELECTION OF VECTOR FIELDS BY USING NEURAL NETWORK TECHNIQUES

Authors: Zhenyu Yang and Benoît Ozell
Submitted to *IEEE VIS 2022* conference in March 2022.

4.1 Abstract

Viewpoint selection is an important task in the domain of scientific visualization and can help scientists easily obtain information from the gathered data. In this article, we introduce a new method that utilizes a neural network to find an ideal viewpoint for the considered data in form of vector fields. Our method constructs a neural network that simulates the dot product between the weights considered as the viewpoint direction and the input vector, and then we train this network by using our newly defined "perpendicularity loss", "same line loss" and "cos distance loss". The normalized vector from the weights with the minimum loss value after training is a good viewpoint direction referring to the criteria represented by the activated loss functions. A final viewpoint is constructed by using the proposed viewpoint generation method. We also present a simple way to build a camera path with spline curves. Our method finds multiple good viewpoints for the tested vector fields in a few seconds for some of our defined use cases and provides trustworthy viewpoints when the same line loss is activated. It is much faster than the previously proposed method using viewpoint entropy.

4.2 Introduction

Scientific visualization is an important tool that helps researchers gather information from the generated 3D data, such as vector fields, streamlines, simple 3D models, and 3D scenes. The vector field is a specific type of data to describe frequently flow movement or natural climate phenomena. Its proper visualization helps to show the essential features for the analysis process, such as the global flow direction, the complexity of each region, and the contour. The automation of visualization generation can effectively accelerate the workflow and save time for exploration. Visualization generation normally involves two tasks: viewpoint selection and camera path generation.

Most viewpoint selection methods [1–7] focus on 3D models or streamlines, rarely on vector fields. And the majority of these methods are based on viewpoint entropy [2, 7] or directly on Shannon entropy. These "entropy" type measures are very dependent on the rendering

process, a small change of one step in the pipeline will potentially cause a modification in the final chosen viewpoint. Our goal is to define a methodology that visualizes vector fields and that is independent from the rendering.

In this paper, we present a viewpoint selection method using neural network techniques that only considers the geometric information provided by the vector field. We construct a single perceptron that takes all vectors in the vector field as inputs and train it with our proposed loss functions which update the weights to satisfy the criterion represented by them. The final viewpoint is constructed with our proposed camera point generation process by considering the normalized minimum weights during the training process as the evaluated viewpoint direction. We also present a simple way to build a camera path from multiple selected camera points by connecting them with spline curves. The method is tested on different vector fields and its results are compared with the viewpoints selected with the traditional viewpoint entropy on the same data set. We also provide two animations for the example-generated camera paths to demonstrate the perception of exploration.

4.3 Related work

Viewpoint selection and camera path generation are two important research fields in computer graphics. Since our paper focuses more on the viewpoint selection task, we explored more previous work about it.

Viewpoint selection

The majority of work about viewpoint selection is based on the concept of Shannon entropy. Vázquez et al. [2, 7] creates the popular "Viewpoint entropy" with it. Its general form is:

$$VE = -\sum_{i=1}^{N_f} \frac{A_i}{A_t} log(\frac{A_i}{A_t}) \tag{4.1}$$

It calculates the weighted average of the information of all visible projected surfaces. It is a general measure for all types of data and generates normally a good camera point in a constant time that depends on the size of the rendered image. Monclús et al. [4] propose another selection measure by modifying the Shannon entropy with the entropy calculated for each RGB channel after the wavelet transform on the original image, this measure is more resistant to noises in the image. These selection methods are all based on single criteria, they deal difficultly with complex 3D models. The multi-criteria selection suggested by Muehler et al. [3] could solve this problem in some scenarios by considering multiple factors in the

selection process, such as viewpoint entropy, occlusion, and the preferred region.

Viewpoint selection for streamlines is frequently more complicated. By applying the Shannon entropy differently, it can be used to calculate the complexity of vectors at a determined position in streamlines, and the selection is realized by picking the viewpoint that generates a view with the maximum complexity [1]. Tao et al. [5] propose a probabilistic-based framework to select viewpoint without the Shannon entropy, good viewpoints are selected by using the mutual information calculated with the related probabilities in one channel of the framework.

Recently, the neural network has become a tool to simulate black-box function and can help to accelerate some calculation processes. Schelling et al. [6] train their network to learn the selection process applied by using different measures, such as viewpoint entropy. Viewpoints with similar quality by the corresponding measure are directly generated through the trained network.

Camera path generation

This task is connected very closely with the graph theory. A camera path can be built by applying a path search algorithm, such as A*, on the constructed graph in the free space of 3D scene [11, 12].

A camera path can also be generated by directly guiding the camera in the movement space through the forces applied by the objects in the scene [13]. This idea of force could be used to refine the camera path considering multiple criteria [11].

The easiest way to construct a camera path is to connect all desired viewpoints by polynomial curves, such as spline curve, Bézier curve, and B-spline. Amamra et al. [14] propose to generate the initial path by using Pythagorean Hodographs Curve and to refine it by keeping away from the bounding volumes of obstacles.

4.4 Methodology

4.4.1 Used simplified notations

We list here the simplified notations that will be used for all mentioned equations in the following subsections:

$$
\begin{gathered}
u = a^2 + b^2, k = a^2 + b^2 + c^2, r = bt_1 - at_2, \\
s = -act_1 - bct_2 + ut_3, n = e_1 b - e_2 a, m = -e_1 ac - e_2 bc + ue_3, \\
A = rm - ns, B = u(kn^2 + m^2), \\
P = norm(\vec{weights}) \cdot \vec{avoidVec}, d = av_1 + bv_2 + cv_3 \\
\textit{The } \boldsymbol{norm} \textit{ in the equation symbolize the normalization operation}
\end{gathered}
\tag{4.2}
$$

4.4.2 Perpendicularity principle

The most important information provided by a vector is its direction. If we want to gather the most amount of information, we should view the vector by a lateral viewpoint. And when our viewing direction is perpendicular to the vector, we can have a complete view of the vector itself and a clear perception of its direction. Since we are viewing a vector field, if we want to obtain the maximum amount of information, the viewing direction should be perpendicular to the majority of the vectors in the vector field. Hence, we can derive the perpendicularity principle: *A viewpoint direction that is perpendicular to the majority of vectors in the vector field provides a viewpoint with the most amount of information of the scene.*

4.4.3 Neural Network

In order to find the direction which is perpendicular to the majority of vectors in the vector field, we construct a single perceptron neural network with an output size of 1 and an input size of 3 as shown in Figure 4.1, and we consider the coordinates of a single vector as one input, and the weights of the perceptron as our viewpoint direction. The activation function of the perceptron is a simple identity function. Thus, the output of this neural network is the dot product of these two vectors.

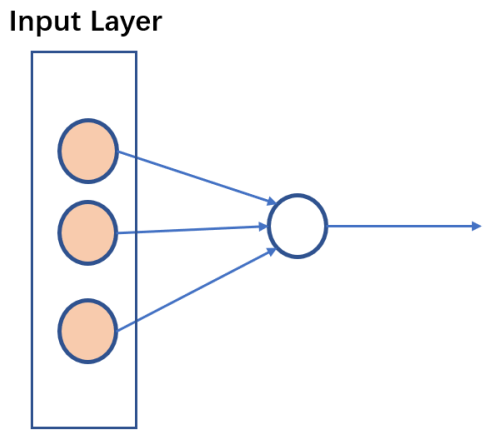


Figure 4.1 Neural network for the prediction of viewpoint direction

4.4.4 Perpendicularity Loss

Definition

Since our objective is to find a direction which is perpendicular to the majority of all vectors, the scalar product of the viewpoint direction and a vector in the scene should be close to 0. And if we train our neural network with a label value of 0 for each vector considered as an input, we can obtain at the end of the training process the weights which have a dot product value close to 0 for each vector, in other words, which is almost perpendicular to all vectors. And these weights can be considered as a viewpoint direction satisfying the perpendicularity principle. Based on this theory, we establish our first loss function which is:

$$L_{pl} = \frac{1}{2N_{vecs}} \sum_{all\ vectors} (label - weights \cdot vector)^2 \tag{4.3}$$

with label = 0 :

$$L_{pl} = \frac{1}{2N_{vecs}} \sum_{all\ vectors} (weights \cdot vector)^2 \tag{4.4}$$

where N_{vecs} is the total number of vectors in the considered vector field.

Derivation

By applying the matrix derivation rules, we obtain its derivative:

$$\frac{\partial L_{pl}}{\partial \vec{weights}} = \frac{1}{N_{vecs}} \sum_{all\ vectors} -(label - weights \cdot vector)vector \tag{4.5}$$

with label = 0 :

$$\frac{\partial L_{pl}}{\partial \vec{weights}} = \frac{1}{N_{vecs}} \sum_{all\ vectors} (weights \cdot vector) vector \tag{4.6}$$

4.4.5 Tornado Problem

The viewpoint direction obtained by using perpendicularity Loss could provide a view with overlap between vectors because of its co-linear and parallel property in the geometric space. A simple example is the "Tornado Model" in Figure 4.2. Some vectors are overlapped because of their parallel and co-linear positions to the projection plan in the view space. The perpendicularity Loss cannot avoid this problem. In order to solve it, we propose our second loss function.

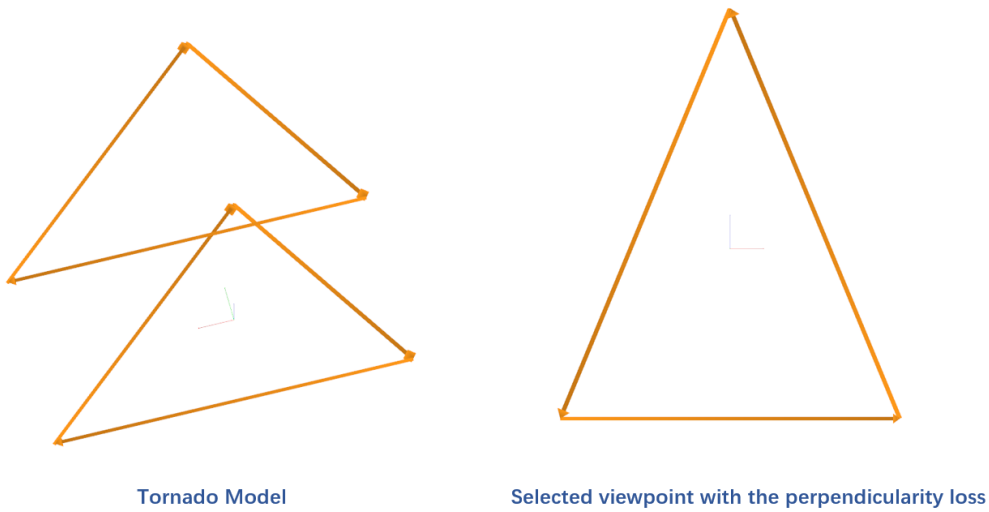


Figure 4.2 Tornado model and the viewpoint found by the perpendicularity loss

4.4.6 Same Line Loss

Definition

Since we want to eliminate the overlap problem raised by the perpendicularity loss, we should create a loss function which will have huge value when vectors are overlapped. For the definition of this function, consider the situation in Figure 4.3.

There are two vectors in our world space, they are separately:

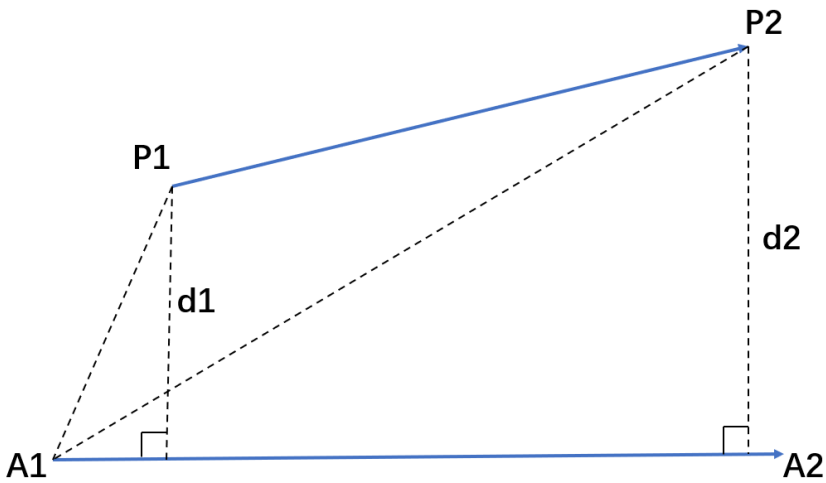


Figure 4.3 Definition of the distance between two vectors

— A vector with the start point P1 and the end point P2.
— A vector with the start point A1 and the end point A2.

We want to calculate the square of the demonstrated distances d1 and d2 on the projection plan in order to define the total distance between these two vectors. Thus, the transformation of these vectors into a view space associated with the viewpoint direction represented by the weights of the network is required. We note the viewpoint direction as:

$$\vec{viewDir} = \vec{weights} = \begin{pmatrix} a \\ b \\ c \end{pmatrix} \tag{4.7}$$

By taking the right vector as the result of the cross product between $\vec{viewDir}$ and z-axis in the world space after the normalization and the up vector as the result of the cross product between $\vec{viewDir}$ and the right vector after the normalization, we form the matrix of transformation for (x,y) coordinates in the desired view space V:

$$\begin{pmatrix} \frac{b}{\sqrt{a^2+b^2}} & \frac{-a}{\sqrt{a^2+b^2}} & 0 \\ \frac{-ac}{\sqrt{(a^2+b^2+c^2)(a^2+b^2)}} & \frac{-bc}{\sqrt{(a^2+b^2+c^2)(a^2+b^2)}} & \frac{a^2+b^2}{\sqrt{(a^2+b^2+c^2)(a^2+b^2)}} \end{pmatrix} \tag{4.8}$$

For the calculation of $d1^2$, we need these vectors:

$$\vec{A1P1} = P1 - A1 = \begin{pmatrix} t_1 \\ t_2 \\ t_3 \end{pmatrix}, \ \vec{A1A2} = A2 - A1 = \begin{pmatrix} e_1 \\ e_2 \\ e_3 \end{pmatrix} \tag{4.9}$$

By applying V to $\vec{A1P1}$ and $\vec{A1A2}$, we can obtain their coordinates on the projection plan in our view space:

$$\begin{aligned} \vec{A1P1}_{view} = V(\vec{A1P1}) &= \begin{pmatrix} \frac{bt_1 - at_2}{\sqrt{a^2+b^2}} \\ \frac{-act_1 - bct_2 + ut_3}{\sqrt{(a^2+b^2+c^2)(a^2+b^2)}} \end{pmatrix} = \begin{pmatrix} \frac{r}{\sqrt{u}} \\ \frac{s}{\sqrt{uk}} \end{pmatrix} \\ \vec{A1A2}_{view} = V(\vec{A1A2}) &= \begin{pmatrix} \frac{be_1 - ae_2}{\sqrt{a^2+b^2}} \\ \frac{-ace_1 - bce_2 + ue_3}{\sqrt{(a^2+b^2+c^2)(a^2+b^2)}} \end{pmatrix} = \begin{pmatrix} \frac{n}{\sqrt{u}} \\ \frac{m}{\sqrt{uk}} \end{pmatrix} \end{aligned} \tag{4.10}$$

And the square of the magnitude of the cross product between $\vec{A1P1}_{view}$ and the unit vector on the direction of $\vec{A1A2}_{view}$ is our desired $d1^2$:

$$d1^2 = (\vec{A1P1}_{view} \times norm(\vec{A1A2}_{view}))^2 = \frac{(rm - ns)^2}{u(kn^2 + m^2)} = \frac{A^2}{B} \tag{4.11}$$

We can apply the same process between the vectors $\vec{A1P2}$ and $\vec{A1A2}$ to obtain the $d2^2$, the total distance between the two vectors in Figure 4.3 is the sum of $d1^2$ and $d2^2$. With this total distance, we define the distance loss between two vectors:

$$L_{distance} = e^{-(d1^2 + d2^2)} \tag{4.12}$$

And by calculating the average of the distance losses for all non-redundant pairs of vectors in the vector field, we define the same line loss:

$$L_{sl} = \frac{1}{N_{pairs}} \sum_{all\ pairs\ of\ vectors} L_{distance} \tag{4.13}$$

where N_{pairs} is the total number for all non-redundant pairs of vectors in the vector field.

The distance loss has a range of value of [0,1], when two vectors get further apart, its value gets closer to the minimum value 0, and when two vectors are overlapped or on the same line, it will have its biggest value 1. With this property, if we train our network to minimize the same line loss, we are forcing all pairs of vectors in the field to keep away from each other and to be not on the same line.

Random version

Sometimes, the same line loss calculated with all pairs of vectors requires a large amount of calculation. To reduce it and accelerate the training process, we randomly pick a small number of vectors from the vector field. And the random version of the same line loss is defined based on these vectors:

$$L_{sl} = \frac{1}{N_{randPairs}} \sum_{pairs\ of\ randomly\ picked\ vectors} L_{distance} \tag{4.14}$$

where $N_{randPairs}$ is the total number for all non-redundant pairs of the randomly picked vectors.

Since these vectors are chosen arbitrarily, they can be considered as the representative vectors for all types of vectors in the vector field. The training with this version of the same line loss can bring a similar effect compared to the complete version, and the required calculation is reduced significantly.

Derivation

By applying the partial derivation rules, we obtain the derivative of $d1^2$ with respect to each coordinate in the weights:

$$\begin{aligned}
\frac{\partial d1^2}{\partial a} &= \frac{2A\frac{\partial A}{\partial a}B - A^2\frac{\partial B}{\partial a}}{B^2} with \\
\frac{\partial A}{\partial a} &= -t_2m - re_1c + 2re_3a + e_2s + nt_1c - 2nt_3a \\
\frac{\partial B}{\partial a} &= 2a(u+k)n^2 - 2ukne_2 + 2am^2 + 2um(-e_1c + 2ae_3) \\
\frac{\partial d1^2}{\partial b} &= \frac{2A\frac{\partial A}{\partial b}B - A^2\frac{\partial B}{\partial b}}{B^2} with \\
\frac{\partial A}{\partial b} &= t_1m - e_2rc + 2e_3br - e_1s + t_2nc - 2t_3nb \\
\frac{\partial B}{\partial b} &= 2b(u+k)n^2 + 2ukne_1 + 2bm^2 + 2um(-e_2c + 2e_3b) \\
\frac{\partial d1^2}{\partial c} &= \frac{2A\frac{\partial A}{\partial c}B - A^2\frac{\partial B}{\partial c}}{B^2} with \\
\frac{\partial A}{\partial c} &= -re_1a - re_2b + nat_1 + nbt_2,\ \frac{\partial B}{\partial c} = 2cun^2 - 2mu(e_1a + e_2b)
\end{aligned} \tag{4.15}$$

By placing the coordinate derivative in a single vector, we obtain the gradient of $d1^2$ with respect to the weights:

$$\frac{\partial d1^2}{\partial \vec{weights}} = \begin{pmatrix} \frac{\partial d1^2}{\partial a} \\ \frac{\partial d1^2}{\partial b} \\ \frac{\partial d1^2}{\partial c} \end{pmatrix} \tag{4.16}$$

The gradient of $d2^2$ with respect to the weights can be constructed with the same approach. Considering these two gradients, the gradient of the distance loss with respect to the weights is:

$$\frac{\partial L_{distance}}{\partial \vec{weights}} = -e^{-(d1^2+d2^2)}(\frac{\partial d1^2}{\partial \vec{weights}} + \frac{\partial d2^2}{\partial \vec{weights}}) \tag{4.17}$$

Finally, we obtain the gradient of the same line loss with respect to the weights:

$$\frac{\partial L_{sl}}{\partial \vec{weights}} = \frac{1}{N_{pairs}} \sum_{all\ pairs\ of\ vectors} \frac{\partial L_{distance}}{\partial \vec{weights}} \tag{4.18}$$

For the random version of the same line loss, it becomes:

$$\frac{\partial L_{sl}}{\partial \vec{weights}} = \frac{1}{N_{randPairs}} \sum_{pairs\ of\ randomly\ picked\ vectors} \frac{\partial L_{distance}}{\partial \vec{weights}} \tag{4.19}$$

Discontinuities

When a vector in the vector field is co-linear with the viewpoint direction, we consider this situation as there is no projection of this vector on the projection plan, and it will not cause any overlap. Hence, the value of a distance loss between this type of vector and other vectors is 0, and the derivative at this point is $\begin{pmatrix} 0 & 0 & 0 \end{pmatrix}^T$.

4.4.7 Cos Distance Loss

Definition

The first two losses are able to determine a single best viewpoint by the criterion, but the loss function proposed in this section helps to find multiple good viewpoints. Firstly, we consider the cos distance between two vectors as the cosine value of the included angle created by these two vectors. By using the cos distance between the viewpoint direction and the vectors (directions) to avoid, we derive the cos distance loss:

$$L_{cd} = \frac{1}{N_{avVecs}} \sum_{all\ vectors\ to\ avoid} -log(1 - (norm(\vec{weights}) \cdot \vec{avoidVec})^2) \tag{4.20}$$

where N_{avVecs} is the total number of vectors to avoid.

All provided vectors to avoid in the equation should be unit vectors.
This loss will grow bigger when the current viewpoint direction get closer to any given vector to avoid, thus, it will force the view direction to search other good viewpoint directions in the rest direction space where some circular cones around the given vectors are removed.

Derivation

We note the normalized weights and a vector to avoid as:

$$norm(\vec{weights}) = \begin{pmatrix} \frac{a}{\sqrt{a^2+b^2+c^2}} \\ \frac{b}{\sqrt{a^2+b^2+c^2}} \\ \frac{c}{\sqrt{a^2+b^2+c^2}} \end{pmatrix} = \begin{pmatrix} \frac{a}{\sqrt{k}} \\ \frac{b}{\sqrt{k}} \\ \frac{c}{\sqrt{k}} \end{pmatrix}, \vec{avoidVec} = \begin{pmatrix} v_1 \\ v_2 \\ v_3 \end{pmatrix} \tag{4.21}$$

By calculating the square of the dot product and applying partial derivative rules on it, we obtain:

$$\begin{gathered} P^2 = (norm(\vec{weights}) \cdot \vec{avoidVec})^2 = \frac{(av_1 + bv_2 + cv_3)^2}{a^2+b^2+c^2} = \frac{d^2}{k}, \\ \frac{\partial P^2}{\partial a} = \frac{2dv_1k - 2d^2a}{k^2}, \frac{\partial P^2}{\partial b} = \frac{2dv_2k - 2d^2b}{k^2}, \frac{\partial P^2}{\partial c} = \frac{2dv_3k - 2d^2c}{k^2}, \\ \frac{\partial P^2}{\partial \vec{weights}} = \begin{pmatrix} \frac{\partial P^2}{\partial a} \\ \frac{\partial P^2}{\partial b} \\ \frac{\partial P^2}{\partial c} \end{pmatrix} \end{gathered} \tag{4.22}$$

Considering this derivative, we apply the derivative rules on the loss function to obtain the final gradient:

$$\frac{\partial L_{cd}}{\partial \vec{weights}} = \frac{1}{N_{avVecs}} \sum_{all\ vectors\ to\ avoid} \frac{1}{1-P^2} \frac{\partial P^2}{\partial \vec{weights}} \tag{4.23}$$

4.4.8 Total Loss Function and Use Cases

Total Loss Function

By combining the mentioned loss functions, we obtain the final used total loss function:

$$L_{final} = L_{pl} + s_{sl}coeff_{sl}L_{sl} + s_{cd}coeff_{cd}L_{cd} \tag{4.24}$$

The new involved parameters are explained below:

— s_{sl} and s_{cd}: The activate states of the same line loss and the cos distance loss (value of 0 or 1).
— $coeff_{sl}$ and $coeff_{cd}$: The weights to amplify or reduce the influence of the same line loss and the cos distance loss.

Use cases

By placing different values for s_{sl} and s_{cd}, we list the common use cases of training:
— $s_{sl} = 0$ and $s_{cd} = 0$: Find a viewpoint direction which is almost perpendicular to all vectors.
— $s_{sl} = 0$ and $s_{cd} = 1$: Find a viewpoint direction which is almost perpendicular to all vectors and not close to the given directions to avoid.
— $s_{sl} = 1$ and $s_{cd} = 0$: Find a viewpoint direction which is almost perpendicular to all vectors and which causes the minimum amount of co-linear projection(avoiding overlap) .
— $s_{sl} = 1$ and $s_{cd} = 1$: Find a viewpoint direction which has the ideal effects of three losses.

4.4.9 Training Method

The neural network is trained with a simple stochastic gradient descendant method by using L_{final}, and each training of the network finds a viewpoint direction to visualize the vector field. The weights with the minimum loss value during the training process are saved and will be used to define the final viewpoint direction, we note it as $\vec{weights}_{min}$. The learning rate and the number of epochs have to be chosen properly considering the provided vector field. The order of magnitude of the number of epochs is usually several hundreds. $coeff_{sl}$ and $coeff_{cd}$ could be estimated with the ratio between the module of their derivative and the module of the perpendicularity loss's derivative at the first epoch if we suppose that all derivatives are in the same order of magnitude, but in order to obtain the best results desired by users, they should be adjusted manually.

4.4.10 GPU Acceleration

The training process is accelerated on the GPU by using multi-thread techniques to calculate in parallel the gradients of multiple inputs for the perpendicularity loss and the same line loss.

4.4.11 Camera Point Generation

Center Point

We pick the center point of a vector field as the average point of all start points of its vectors. We don't use the end point because a vector with a long magnitude could drag the center point to its direction, and stay away from the majority of all vectors. We note the center point as cp.

Minimum Distance To The Center Point

We consider the set of the start points and the end points of all vectors in the vector field as its associated point cloud. And we define the projected distance of a point on the viewpoint direction to the center point as:

$$projDistance_{point} = norm(\vec{weights}_{min}) \cdot (point - cp) \tag{4.25}$$

We take the minimum distance of the projected distances of all points in the point cloud as the minimum distance to the center point, we note this distance as $distanceTocp_{min}$.

Camera Point

The final constructed camera point associated with the trained viewpoint direction is:

$$\begin{gathered} \vec{viewDir}_{final} = norm(\vec{weights}_{min}), \\ cameraPos_{final} = cp + (distanceTocp_{min} - cstDist)\ \vec{viewDir}_{final} \end{gathered} \tag{4.26}$$

We take the unit vector in the direction of $\vec{weights}_{min}$ as the final viewpoint direction and the position keeping away from the center point by following the final viewpoint direction with the demonstrated distance as the camera point position. $cstDist$ is a positive constant value that we reduce to keep away from the center point for the orthogonal projection.

Choice of Projection Matrix

In order to simplify the rendering process, we choose the orthogonal projection. We take a view space as the coordinates system constructed by these vectors:

- $\vec{viewDir}_{final}$
- $\vec{up_{zorx}}$: The unit vector in the direction of the cross product between $\vec{viewDir}_{final}$ and the z-axis(if $\vec{viewDir}_{final}$ is co-linear with or close to the z-axis, we change this axis

to x-axis)

— $\vec{right}_{zorx}$: The unit vector in the direction of the cross product between $\vec{viewDir}_{final}$ and $\vec{up}_{zorx}$

We note the view matrix of this view space as $MView_{zorx}$. By using it, We place the point cloud associated with the vector field into the considered view space. Then, these needed values is calculated in the range of all points:

— Width: The maximum of the absolute values of x coordinates.

— Height: The maximum of the absolute values of y coordinates.

— Near: The minimum of the absolute values of z coordinates.

— Far: The maximum of the absolute values of z coordinates.

The chosen final projection matrix is the matrix associated with the projection of (-width, width, -height, height, near, far + ϵ). ϵ is a positive constant that we add to the "far" position to ensure the correct projection when the values of near and far are too close.

4.4.12 Even Function Property

We can easily observe and prove that L_{final} is an even function in every use case (in other words, $L_{final}(\vec{weights}) = L_{final}(-\vec{weights})$). With this property, when a good viewpoint direction is found by using our neural network, its inverse direction is also a direction with the same quality by the criterion of L_{final}. Thus, we can obtain two good camera points with a single training process by constructing camera points from the determined viewpoint direction and its inverse direction.

4.4.13 Camera Path Generation

Generation

With a series of camera points found by using our neural network, we can build a camera path by connecting each two camera point from the start viewpoint to the end viewpoint with a spline curve of degree 3. We define the path direction between two considered camera points as $\vec{pathDir} = norm(endCameraPoint - startCameraPoint)$. In order to generate the spline curve between two camera points, we calculate these vectors for each camera point:

$$\vec{up} = \vec{pathDir} \times \vec{viewDir}, \vec{right} = \vec{viewDir} \times \vec{up} \tag{4.27}$$

$\vec{up}$ helps to define the orientation of view at this viewpoint, and $\vec{right}$ is used as the direction vector at this point to generate the spline curve. Thus, the final constructed camera path

between these camera point is:

$$\begin{aligned} P(t) = H_0(t)cameraPos_{startPoint} + H_1(t)cameraPos_{endPoint} \\ +H_2(t)\vec{right}_{startPoint} + H_3(t)\vec{right}_{endPoint} \end{aligned} \tag{4.28}$$

$H_0(t), H_1(t), H_2(t), H_3(t)$ are Hermite polynominals of degree 3. We inject different values of t into the path function to obtain the camera point associated with this t value on the constructed camera path, its $\vec{up}$ for the orientation of view and its viewpoint direction are calculated by interpolating the corresponding vectors at the start point and the end point with its t value:

$$\begin{aligned} \vec{up}_t = (1-t)\vec{up}_{startPoint} + t\vec{up}_{endPoint} \\ \vec{viewDir}_t = (1-t)\vec{viewDir}_{startPoint} + t\vec{viewDir}_{endPoint} \end{aligned} \tag{4.29}$$

Continuity Problem

The middle point of two consecutive spline curves will have two $\vec{up}$ calculated for each spline curve. In order to guarantee the continuity of display, we take $\vec{up}$ of the first spline curve as its orientation of view.

4.5 Results and discussions

4.5.1 Test method

To verify the effect of our viewpoint selection method, we have tested it on five different data sets, they are separately:

- Cylinder: multiple vector groups that form a symmetric cylinder.
- Asymmetric cylinder (A-Cylinder): multiple vector groups that form an asymmetric cylinder.
- Industrial furnace burner (Burner) : A vector field describing the velocity field of the injected fuel.
- Train: A vector field describing the airflow that collides with the train.
- Aneurysm: A vector field describing the blood movement in the vicinity of the brain aneurysm.

For each data set, we have applied two groups of tests to obtain the generated viewpoints for all possible use cases of L_{final}. The first group contains these steps:

- One training with $s_{sl} = 0$ and $s_{cd} = 0$: Obtain the viewpoint generated by the perpendicularity loss.

— One training with $s_{sl} = 1$ and $s_{cd} = 0$: Obtain the viewpoint generated by the perpendicularity loss and the same line loss.

— Five trainings with $s_{sl} = 0$ and $s_{cd} = 1$: Obtain multiple viewpoints generated by the perpendicularity loss and the cos distance loss. For the beginning of this step, the two viewpoint directions in the first two steps are considered as the original directions to avoid. After each training, the newly generated viewpoint direction is added to the group of directions to avoid.

The steps of the second group are:

— One training with $s_{sl} = 0$ and $s_{cd} = 0$: Obtain the viewpoint generated by the perpendicularity loss.

— Three trainings with $s_{sl} = 1$ and $s_{cd} = 1$: Obtain multiple viewpoints generated by all three losses. The viewpoint direction found in the first step is the first direction to avoid. The newly generated direction at each training will be avoided by the next training by adding it to the group of directions to avoid.

For comparison, we have run on all data sets an algorithm that found the viewpoint with the highest viewpoint entropy value of 382 candidates on the bounding sphere around the vector field. The evaluation of the viewpoint entropy for each candidate was calculated with a window size of 900×600 pixels. Every selected "entropy" viewpoint is considered as the reference to other viewpoints for each data set.

The camera path generation method has been tested on the "Burner" and "A-Cylinder" data set. For each data set, the method was tested with four viewpoints found by only the perpendicularity loss and its combination with the cos distance loss. An animation of the navigation on the built camera path has also been generated for each data set.

4.5.2 Viewpoint selection results

General functionality

To demonstrate the general functionality of our viewpoint selection method, we choose the results obtained on the "Asymmetric cylinder" data set.

We firstly analyse the results obtained in the first group of tests, as shown in Figure 4.4. We can observe that the viewpoint (b) provides a view with a large amount of perpendicularity to the vectors, this justifies the network has chosen the viewpoint direction by the criterion of the perpendicularity loss, and this loss worked as expected. But, there is a lot of overlap in the viewpoint (b). The viewpoint (c) retains a good amount of perpendicularity, and avoids the overlap between the vectors, each vector is displayed more individually. This viewpoint

validates the good functioning of the same line loss among the perpendicularity loss. The viewpoints (d-e-f-g-h) show the effort of the cos distance loss. It is trying to find different viewpoints than the previously found viewpoints and to maintain the perpendicularity at the same time. These viewpoints prove that the cos distance loss works well as designed. However, its performance is not always stable, for example, the viewpoint (g) is with less good quality and many vectors are overlapped. And after multiple trainings with the cos distance loss, the remaining free space for the exploration is very limited, the cos distance loss will struggle to find a new viewpoint while guaranteeing the perpendicularity.

By comparing the viewpoints found by our network to the "entropy" viewpoint (a), we discover that the first found viewpoint (b) with only the perpendicularity loss has already similar quality to the reference viewpoint, but there is a lot of overlap between vectors. By activating the same line loss in L_{final}, the second found viewpoint (c) could be described as having better quality than (a) because the structure of the bottom group of four sets of vectors is more visible. Some of the viewpoints generated by the combination of the cos distance loss and the perpendicularity loss have close quality compared to (a), but, the rest of them have certain differences in quality from it.

The viewpoints generated in the second group of tests are shown in Figure 4.5. We can see that the viewpoints (c-d-e) generated by all three losses are different, and they retain a good amount of perpendicularity. (We also observed this property for other models.) There is also a small amount of overlap in them compared to (b). This phenomenon proves that the all-activated form of L_{final} correctly trains the network to find a viewpoint direction different from previously found ones while retaining the perpendicularity and avoiding the overlap between vectors. By comparing the viewpoints (c-d-e) to the reference viewpoint (a), we discover that they all have good quality as the viewpoint found by the viewpoint entropy. This point justifies the stability of the quality of the found viewpoints by all three losses.

Verification of the even function property

As mentioned in the section subsection 4.4.12, one training of our network can find two viewpoint directions with the same quality by the criterion of L_{final}. By looking at the pairs of viewpoints in our results ((c)(d), (e)(f) in Figure 4.6 and (b)(c), (d)(e) in Figure 4.8), we can observe that viewpoints in each pair have similar visual quality in many aspects, such as the number of pixels, the level of overlap, and the perpendicularity. This point verifies the even function property on the visual level.

Analysis of the perpendicularity loss on other models

If we take all viewpoints generated by only the perpendicularity loss (every (b) viewpoint in Figure 4.6, Figure 4.7 and Figure 4.9, and (b)(c) in Figure 4.8), we discover that all viewpoints are also very perpendicular to the corresponding vector field. This could be a good effect in some cases, for example, the inverse viewpoints (c) in Figure 4.8 is almost identical to the "entropy" viewpoint, and displays a large amount of pixels. The viewpoint (b) in Figure 4.7 is another good example, it shows more completely the flow movement of blood than the reference viewpoint (a). However, in Figure 4.6 and Figure 4.9, the viewpoints generated by the perpendicularity provide a lot of overlap between vectors, and this disturbs the perception of the overall structure of vector fields.

Analysis of the same line loss on other models

To analyse more independently the effect of the same line loss, we look at the results obtained only with the combination of the perpendicularity loss and the same line loss, they are:

— Every (c) in Figure 4.7, and Figure 4.9.
— (c)(d) in Figure 4.6.
— (d)(e) in Figure 4.8.

By comparing these viewpoints to the corresponding viewpoints generated by only the perpendicularity loss in each data set, we can conclude with more confidence that the same line loss helps to reduce the overlap and that it displays more individually the vectors. This type of display provides nearly identical or better results in most cases than the "entropy" viewpoint. For example, the viewpoint (c) in Figure 4.9 displays more clearly the three-dimensional structure of the vector field, and the (c) in Figure 4.7 is very alike as the "entropy" viewpoint. But, for the model "Burner" in Figure 4.6, the viewpoints generated with the same line loss still covers a part of the structure, and is not good as the "entropy" viewpoint.

An interesting result is the one for the "Train" model in Figure 4.8. The viewpoint (e) shows more clearly the expansion of the airflow caused by its colliding with the train, but, it contains less pixels. We cannot simply conclude it as a good or bad viewpoint. This depends on the user's intention.

Analysis of the cos distance loss on other models

Same as the analysis method used in the previous section, we use only the results with the perpendicularity loss and the cos distance loss. They are:

— (e-f-g-h-i-j) in Figure 4.6.
— (d-e-f-g-h) in Figure 4.9.

Referring to these results, the effect of finding new viewpoints by avoiding previously generated viewpoints is verified for the cos distance loss.

The viewpoints in the "Burner" model are mostly with good quality because of the clarity of the orientation of the vectors. Especially the viewpoint (e), it is almost identical to the "entropy" viewpoint which is a nicely chosen viewpoint for this model. But, we cannot confirm its stability considering the results in Figure 4.4.

The viewpoints in the "Cylinder" model have mostly similar quality as the reference viewpoint considering the three-dimensional perception provided by them. They also show the struggle of the cos distance loss when the vector field has a symmetric form. It is just turning around the model to find new viewpoints.

Magnitude of run time

All defined tests were performed on a GeForce GTX 1060 graphic card. The time results are shown in Table 4.1, the unit for all displayed run times is milliseconds. If we have run multiple trainings for one test case, the displayed time is the average time through all tests. To simplify notations, we abbreviate the perpendicularity loss as pl, the same line loss as sl, the cos distance loss as cd, all three losses as all and the viewpoint entropy as VE.

We can see that the evaluation using the viewpoint entropy spends a constant time since the test algorithm only depends on the window size. The magnitude of time for our method is in the range of a few seconds when there is not the same line loss. The presence of the same line loss raises the run time to the range of 30 seconds to 150 seconds, the actual run time depends on the data set. However, the run time for the viewpoint entropy method is close to 300 seconds, this is way beyond our method. Even if we sum up the run times in the first group of tests for any data set, the total time is less than the run time of the viewpoint entropy method.

If the viewpoint generated by only the perpendicularity loss has already good quality, our method finds a valuable viewpoint by being hundreds of times faster than the viewpoint entropy method. Even if the first one is not a very considerable viewpoint, we can train our network by activating the same line loss to find a trustworthy viewpoint. If the user thinks the training process with the same loss is too slow, he can activate just the cos distance loss to generate multiple possible good viewpoints and selects the ones that conform to his intention, the run time for this process could spend only a few seconds for a not very big vector field.

The user can train the network with all three losses multiple times to find multiple good viewpoints with stable quality if he wants.

4.5.3 Camera path generation

For each tested data set, we have saved the viewpoints using to construct the camera path and the generated camera path itself. The results are displayed in Figure 4.10 and Figure 4.11. The blue line displayed in (e) for each figure is the generated camera path, and the magenta vectors are the viewpoint directions for each camera point.

By observing the path for the "Burner" model in Figure 4.10,We discover that the generated path goes through the vector field. This may not be so convenient if the user wants to see the complete vector field during the movement.

By looking at the path for the "Asymmetric cylinder" model in Figure 4.11, we can see the curvature provided by connecting the viewpoints by spline curves. However, the curvature is not so visible for the "Burner" model in Figure 4.10.

The generated animations for the built camera paths of these two data sets are provided in the videos accompanying this paper. These animations show the navigation of the camera following the corresponding generated path and oriented using the up vector calculated with the formulas mentioned in section 4.4.13. The dynamic perception and the continuity of motion presented in these animations justify that the paths generated using our method are generally acceptable, and could be considered good paths for viewing the vector fields.

Data set	pl	pl and sl	pl and cd	all	VE
Cylinder	1497	80677	1571	81716	247438
A-Cylinder	1270	65057	1434	65160	258790
Burner	4488	76098	4549	73731	275850
Train	5370	145172	5543	146639	258455
Aneurysm	2395	37360	2491	37071	235573

Table 4.1 Average run times (in milliseconds) for all defined tests

4.6 Future works

Our network is still very simple. If we want to extend the functionality of the network or improve its performance, the invention of new loss functions or the extension of the network structure is needed.

A selection step of representative vectors in viewpoints could be added to our method to display viewpoints more clearly.

The learning rate selection process could be automatized by an algorithm that tests a group of learning rates with different orders of magnitude and preserves the first one that allows the network to begin the learning process. The determination of the number of epochs could be eliminated if the end of training is automatically detected by the small changes in loss difference.

Our viewpoint selection method could potentially be applied in normal 3D models if the skeleton in form of vectors can be generated. This generation process is possible by using the medial axis algorithm mentioned in [11].

4.7 Conclusion

We proposed an innovative approach to select good viewpoints by training a single perceptron with the geometric information in vector fields, and a simple way to generate camera paths based on multiple viewpoints with spline curves. The introduced new viewpoint selection method can generate one viewpoint in about 5 seconds and multiple good viewpoints within about 30 seconds by using the perpendicularity loss and its combination with the cos distance loss. This clearly outperforms the existing method with the viewpoint entropy by at least ten times faster. Furthermore, this method can find viewpoints with less overlap by activating the same line loss, and multiple trustworthy viewpoints can be determined when all three losses are applied.

We believe in the future expansion of the application of our method when a skeleton in form of vectors for normal 3D models or 3D scenes can be simply generated.

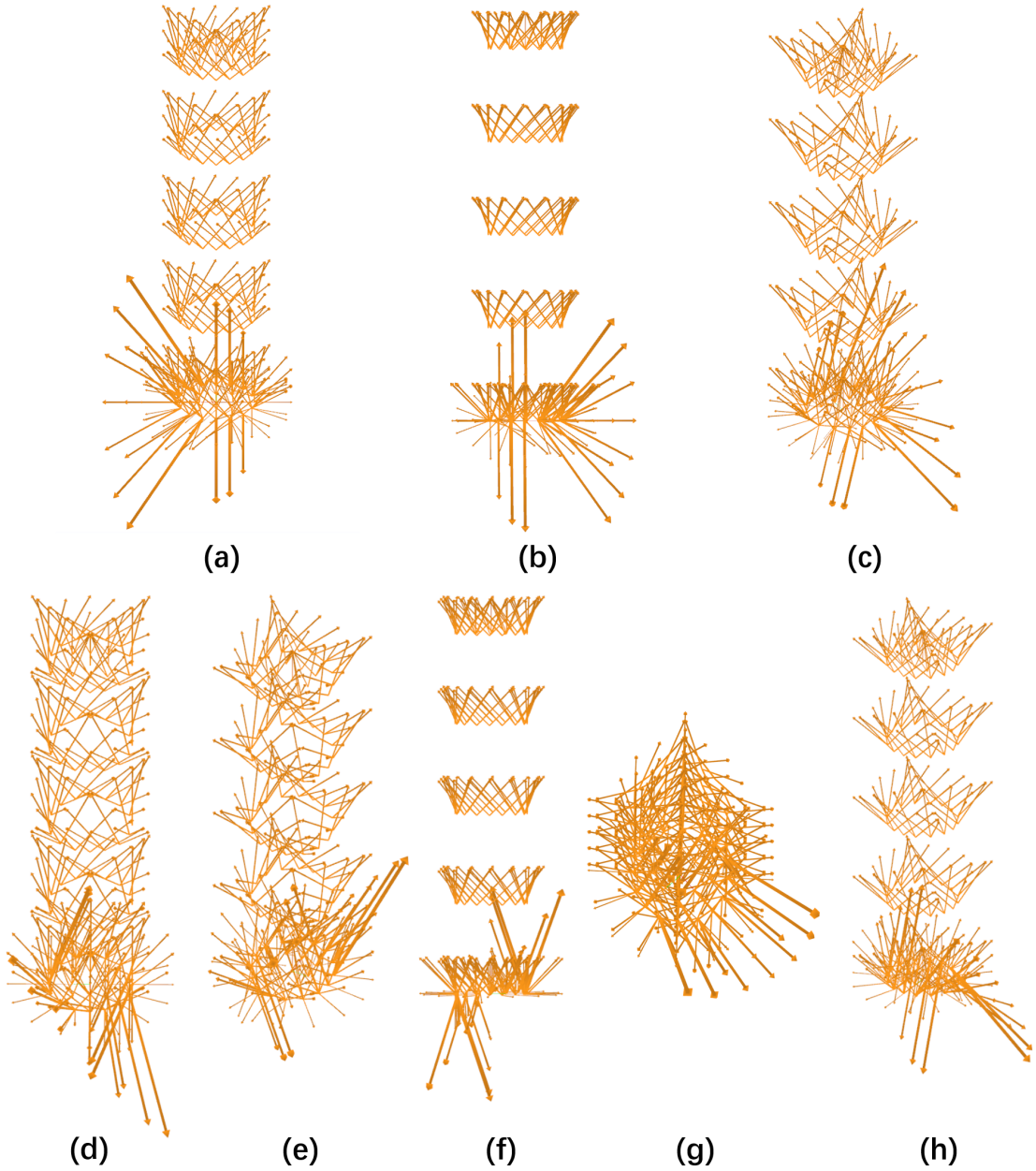


Figure 4.4 Generated viewpoints for the "Asymmetric cylinder" in the first group of tests. The number of vectors in this model is 506. The same line loss used in this model is the complete version. (a) is the reference "entropy" viewpoint. (b) is the viewpoint generated by the perpendicularity loss. (c) is the viewpoint generated by the combination of the perpendicularity loss and the same line loss. (d-e-f-g-h) are the viewpoints generated with the perpendicularity loss and the cos distance loss. All viewpoints generated with our network are obtained by using a learning rate of 0.1. The number of epochs is 600 for (b-d-e-f-g-h) and 200 for (c).

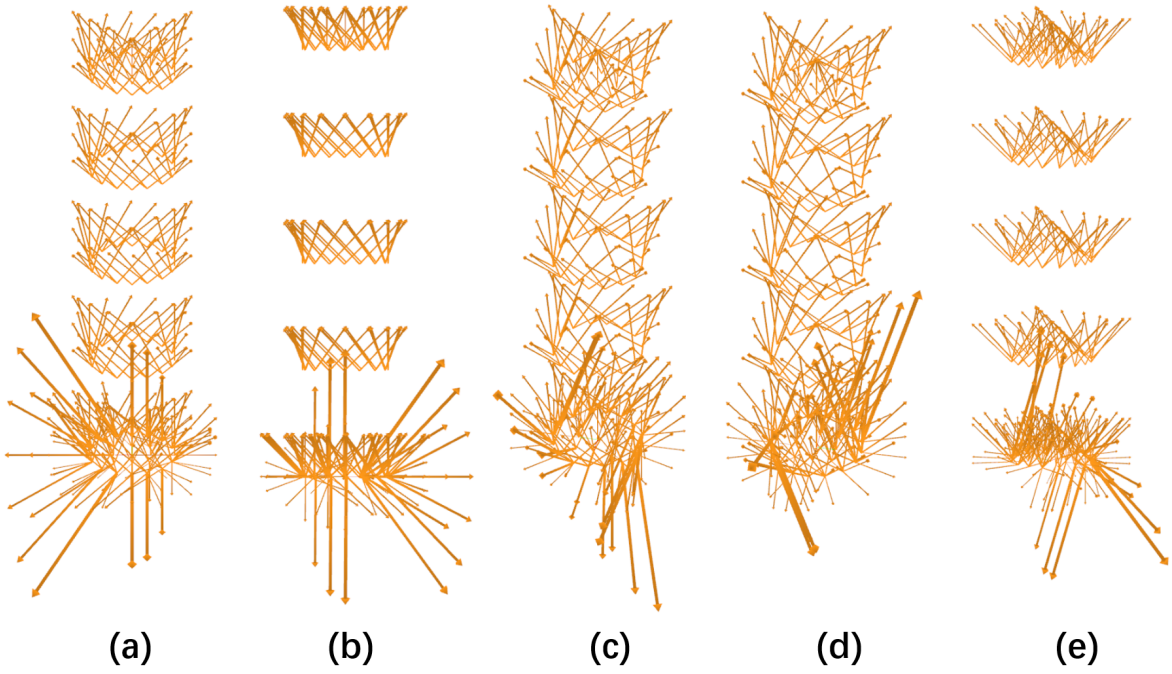


Figure 4.5 Generated viewpoints for the "Asymmetric cylinder" in the second group of tests. The number of vectors in this model is 506. The same line loss used in this model is the complete version. (a) is the reference "entropy" viewpoint. (b) is the viewpoint generated by the perpendicularity loss. (c-d-e) are the viewpoints generated with all three losses. All viewpoints generated with our network are obtained by using a learning rate of 0.1. The number of epochs is 600 for (b) and 200 for (c-d-e).

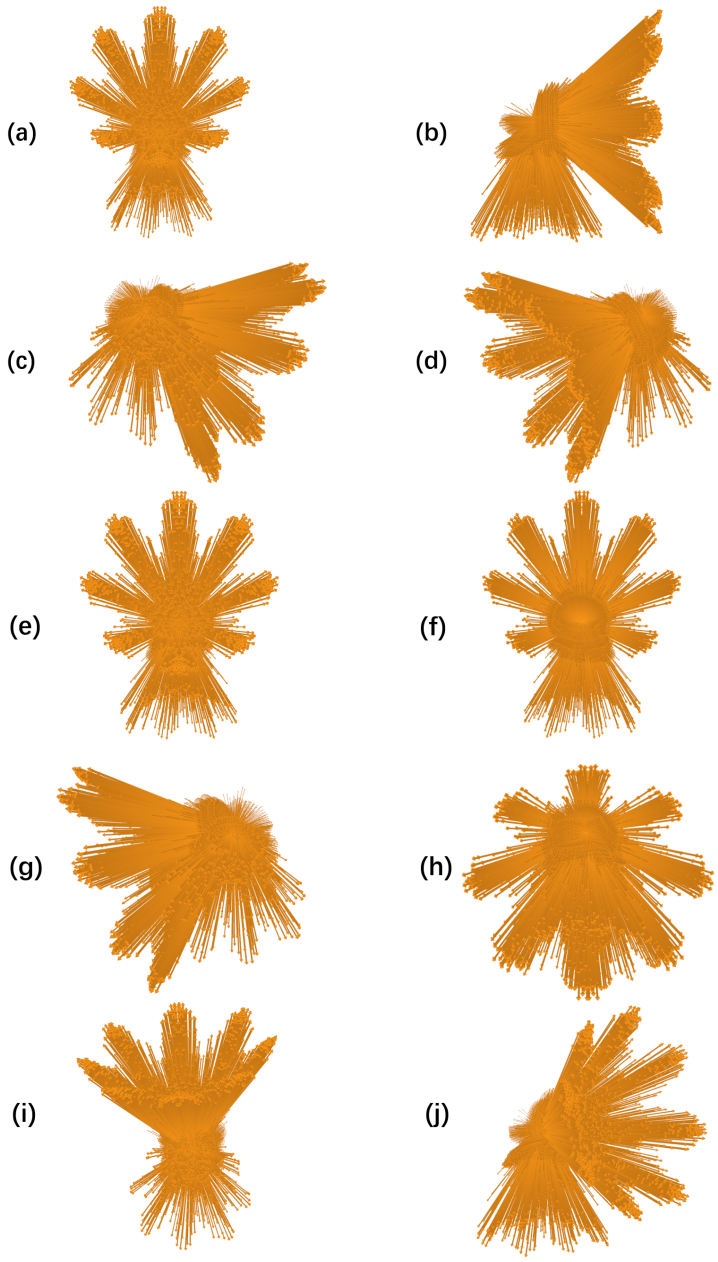


Figure 4.6 Generated viewpoints for the "Burner" in the first group of tests. The number of vectors in this model is 49929. The same line loss used in this model is the random version, the number of randomly picked vectors is 1000. (a) is the reference "entropy" viewpoint. (b) is the viewpoint generated by the perpendicularity loss. (c)(d) are the viewpoints generated by the perpendicularity loss and the same line loss. (e-f-g-h-i-j) are the viewpoints generated by the perpendicularity loss and the cos distance loss. (d) and (f) are the viewpoints obtained directly by the inverse directions of (c) and (e) based on the even function property. All viewpoints generated with our network are obtained by using a learning rate of 0.00005. The number of epochs is 100 for (c)(d) and 200 for (b-e-f-g-h-i-j).

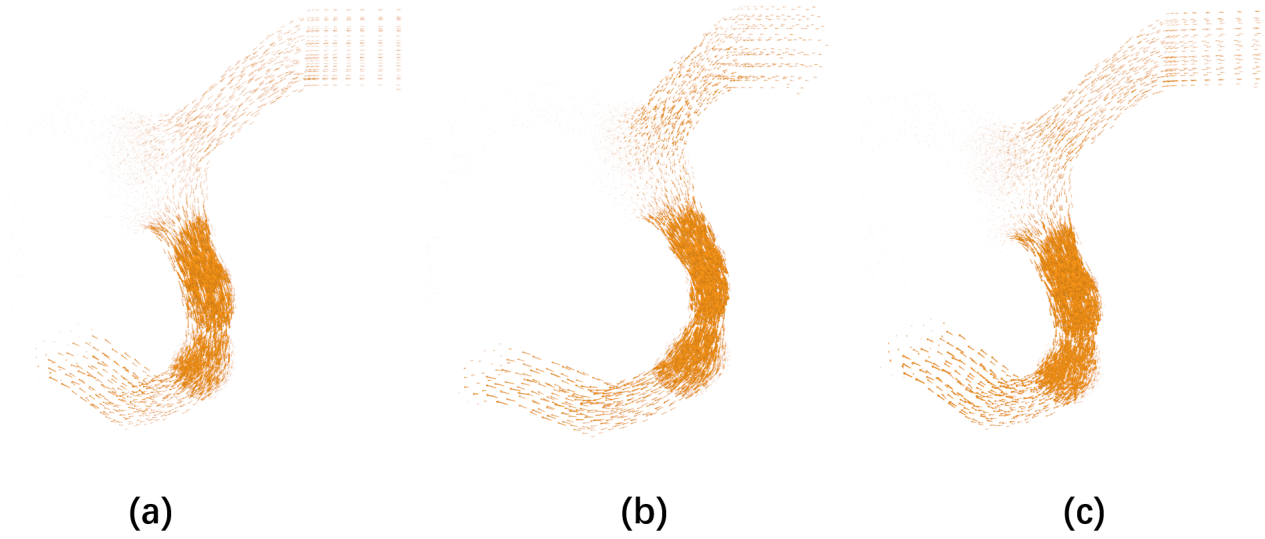


Figure 4.7 Some generated viewpoints for the "Aneurysm" in the first group of tests. The number of vectors in this model is 23288. The same line loss used in this model is the random version with the number of randomly picked vectors of 1000. (a) is the reference "entropy" viewpoint. (b) is the viewpoint generated by the perpendicularity loss. (c) is the viewpoints generated by the perpendicularity loss and the same line loss. The learning rate is 1 for all training process. The number of epochs is 200 for (b) and 50 for (c).

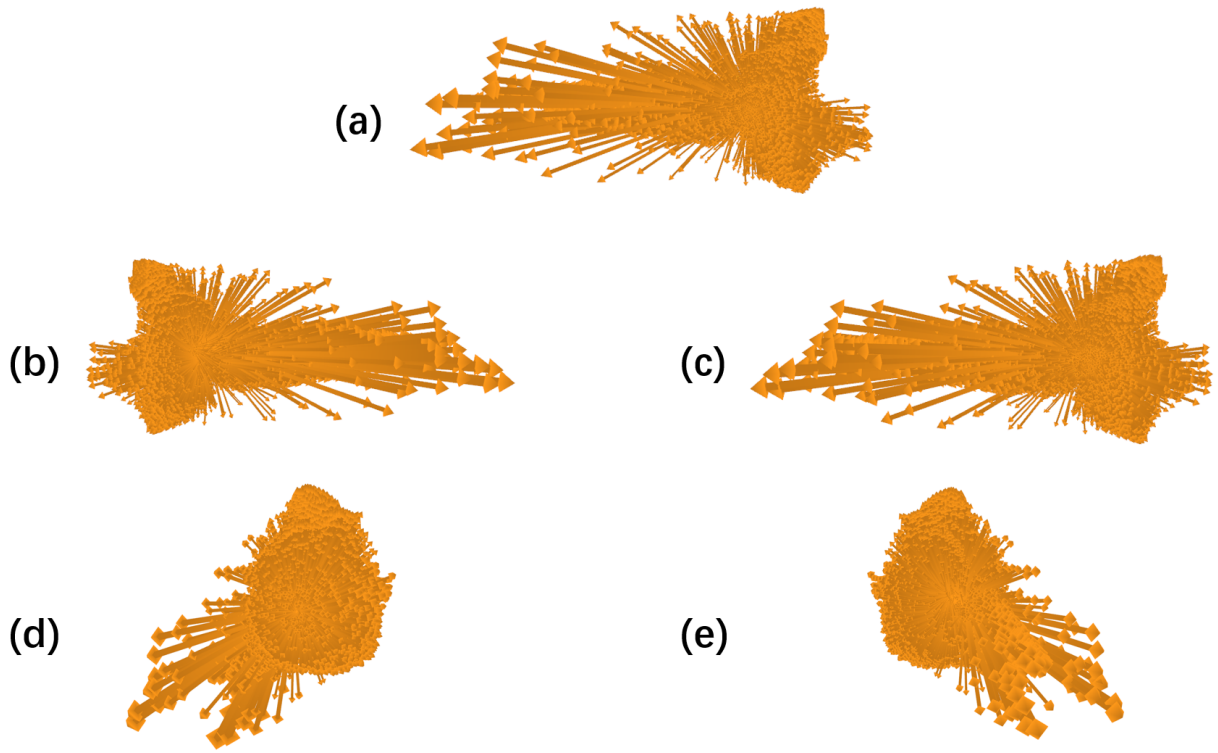


Figure 4.8 Some generated viewpoints for the "Train" in the first group of tests. The number of vectors in this model is 12071. The same line loss used in this model is the random version with the number of randomly picked vectors of 1000. (a) is the reference "entropy" viewpoint. (b)(c) are the viewpoints generated by the perpendicularity loss. (d)(e) are the viewpoints generated by the perpendicularity loss and the same line loss. (c) and (e) are the viewpoints obtained directly by the inverse directions of (b) and (d) based on the even function property. The learning rate is 0.0005 for all training process. The number of epochs is 800 for (b)(c) and 200 for (d)(e).

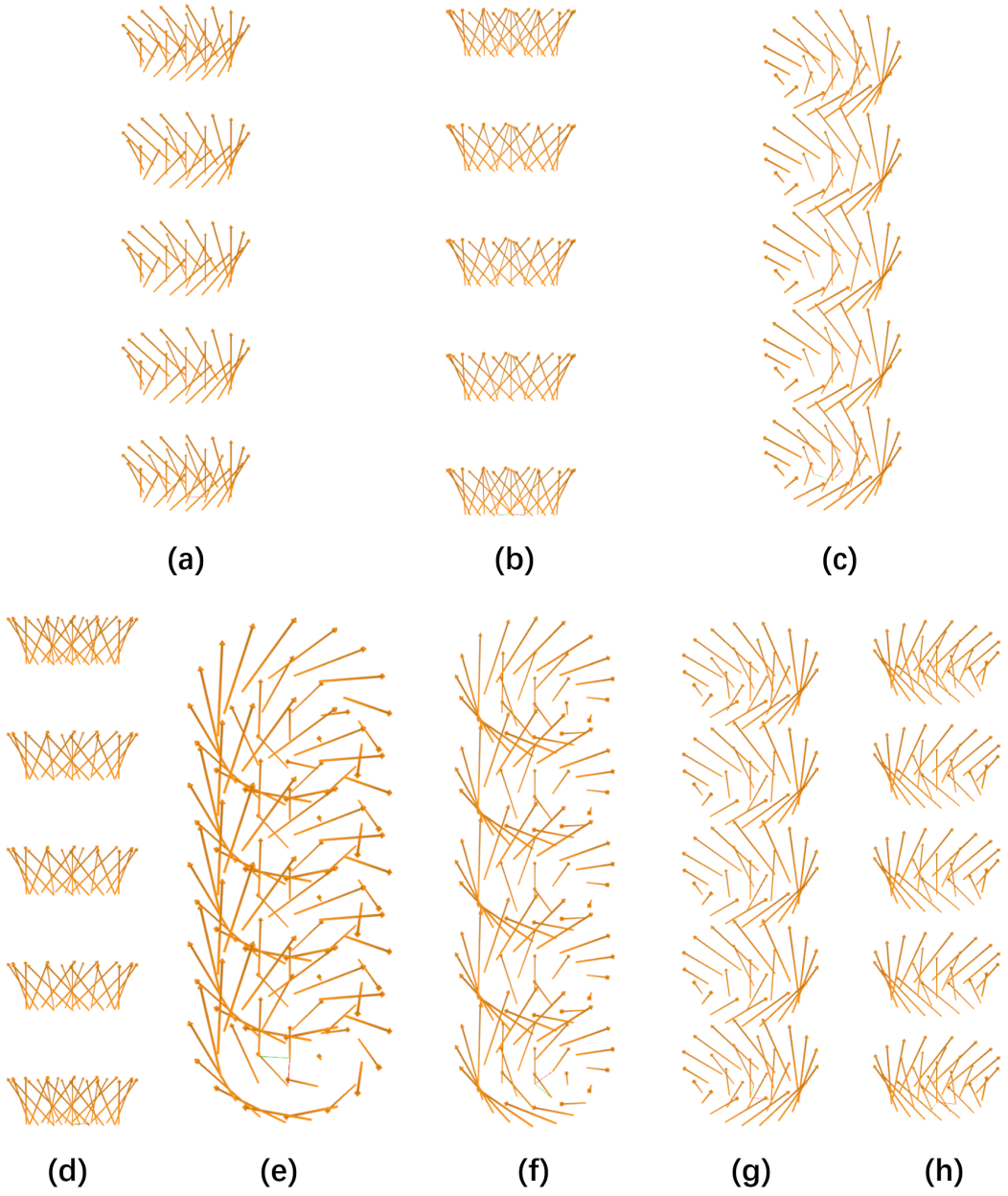


Figure 4.9 Generated viewpoints for the "Cylinder" in the first group of tests. The number of vectors in this model is 221. The same line loss used in this model is the complete version. (a) is the reference "entropy" viewpoint. (b) is the viewpoint generated by the perpendicularity loss. (c) is the viewpoint generated by the combination of the perpendicularity loss and the same line loss. (d-e-f-g-h) are the viewpoints generated with the perpendicularity loss and the cos distance loss. All viewpoints generated with our network are obtained by using a learning rate of 0.05. The number of epochs is 700 for (b-d-e-f-g-h) and 600 for (c).

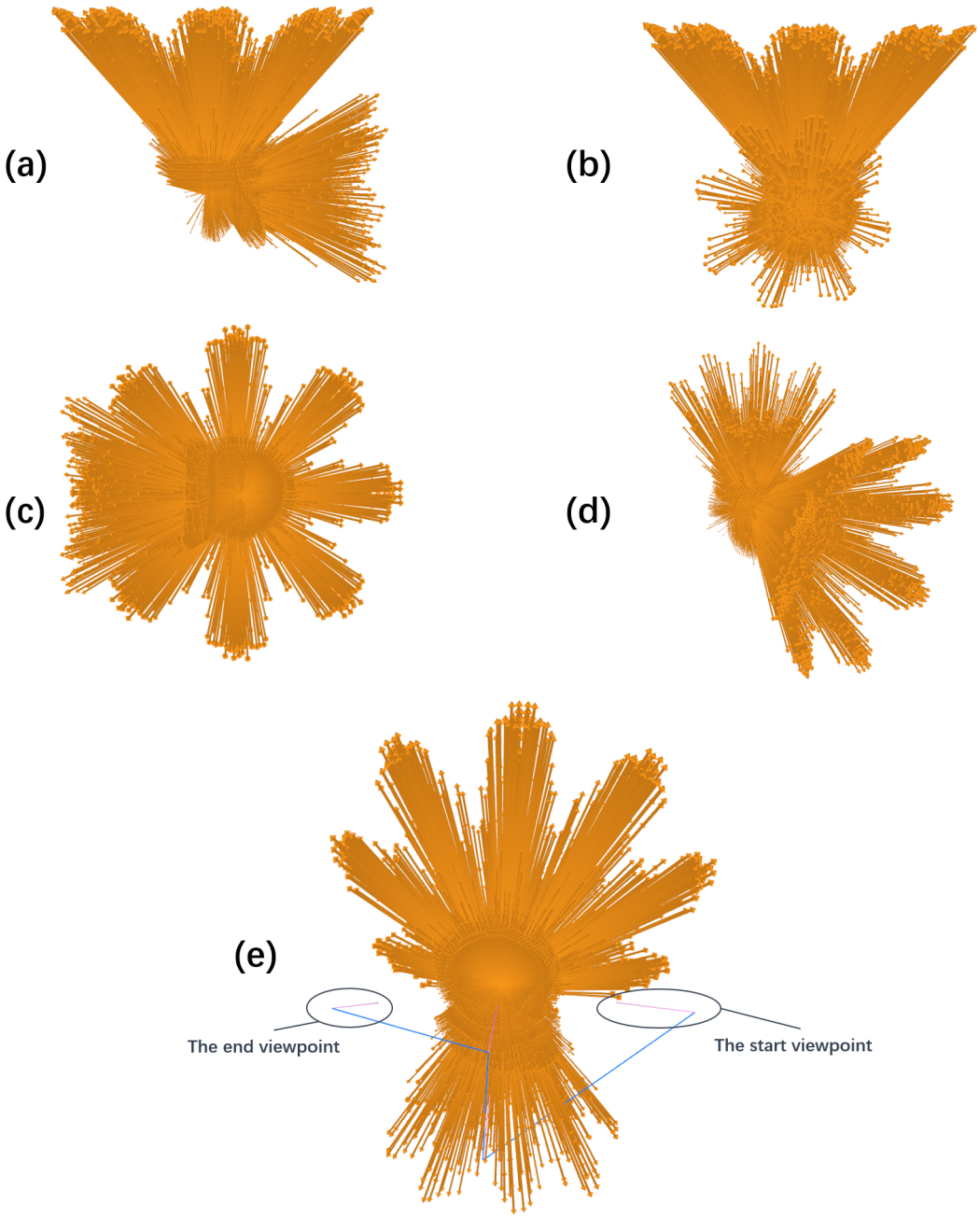


Figure 4.10 Generated camera path for the "Burner" and the viewpoints used to construct this path. (a) is the viewpoint generated by the perpendicularity loss. (b-c-d) are the viewpoints generated by the perpendicularity loss and the cos distance loss. (e) is the generated camera path by the presented viewpoints. (a) is the start viewpoint of the path and (d) is the end point.

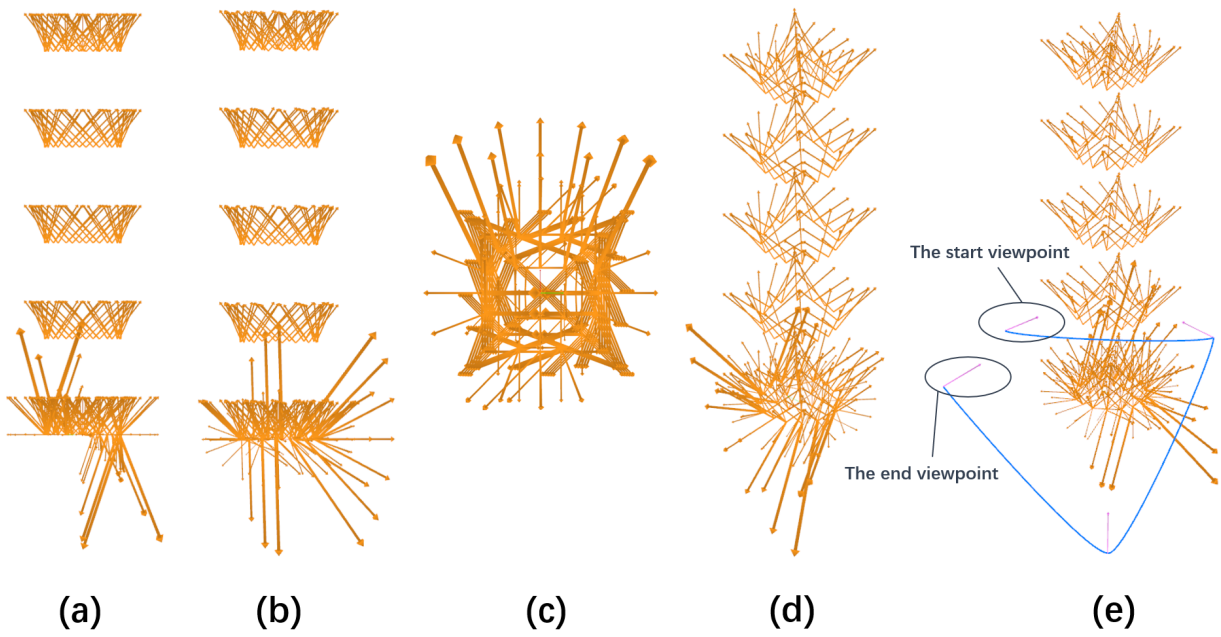


Figure 4.11 Generated camera path for the "Asymmetric cylinder" and the viewpoints used to construct this path. (a) is the viewpoint generated by the perpendicularity loss. (b-c-d) are the viewpoints generated by the perpendicularity loss and the cos distance loss. (e) is the generated camera path by the presented viewpoints. (a) is the start viewpoint of the path and (d) is the end point.

CHAPITRE 5 DISCUSSION GÉNÉRALE ET COMPLÉMENTAIRE

Dans cette section, nous discutons de l'innovation de notre méthodologie et des parties complémentaires qui ne sont pas analysées dans l'article. Ces parties sont :

— Instabilité inhérente de la méthode utilisant l'entropie de point de vue.

— Méthode précise utilisées pour accélérer l'entraînement des fonctions de perte de la perpendicularité et de la colinéarité (*en anglais, the perpendicularity loss and the same line loss*).

— Algorithme pour générer une animation de la navigation sur un chemin de caméra construit.

5.1 Innovation de notre méthodologie

La partie la plus novatrice de notre méthodologie est celle sur la sélection des points de vue. De nombreuses méthodes antérieurement proposées ([1–4, 6, 7]) utilisent l'entropie de Shannon afin de construire leur mesure pour la sélection des points de vue. Même celle qui se base sur des théorèmes de Bayes ([5]) utilise la théorie de l'information qui est liée à l'entropie de Shannon pour définir son critère de sélection. De plus, les méthodes concernant l'entropie de Shannon dépendent toujours du rendu et ce point est un facteur instable pour la performance.

Notre méthode pense la question différemment en se focalisant plutôt sur les informations géométriques portées par le champ vectoriel et est indépendante du rendu. De plus, l'application des techniques du réseau de neurones est aussi un aspect innovant de notre méthode par rapport à une majorité des méthodes précédentes.

5.2 Instabilité de l'entropie de point de vue

Nous avons mentionné plusieurs fois la dépendance entre l'entropie de point de vue et la méthode de rendu. Nous présentons ici un exemple dans la Figure 5.1. Les deux points de vue montrés dans la figure sont obtenus en réalisant l'évaluation selon l'entropie de point de vue : (a) est celui qui est obtenu quand nous affichons les vecteurs avec une tête plus grande, (b) est celui avec des têtes affichées plus petites. Nous observons un renversement du champ vectoriel et des rotations de l'angle d'observation. Ce changement important est provoqué simplement par une petite modification dans la méthode de rendu. Ce exemple démontre bien l'instabilité de l'entropie de point de vue.

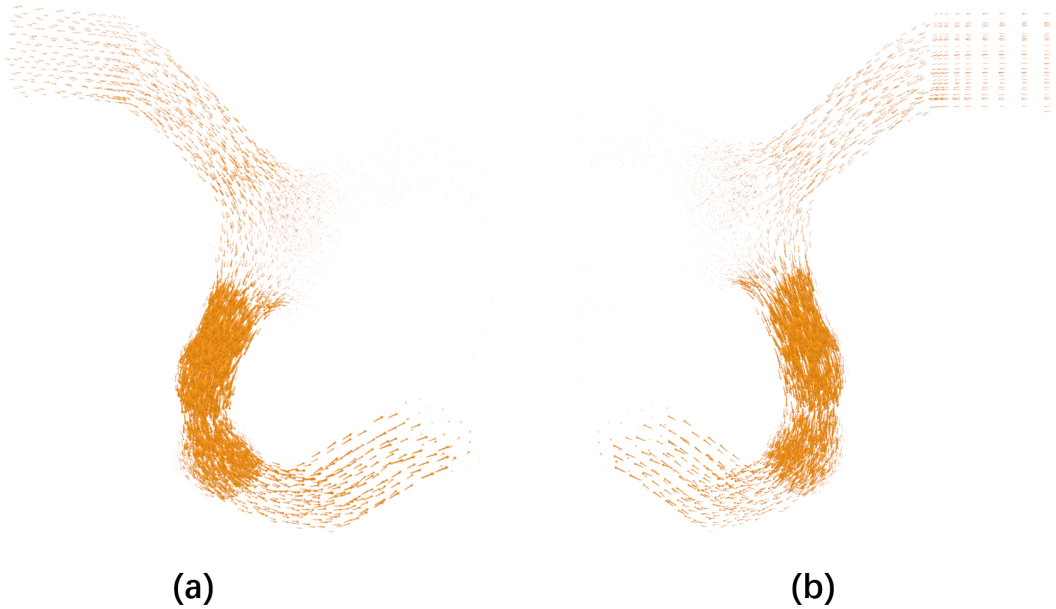


Figure 5.1 Deux points de vue générés par l'entropie de point de vue. (a) est celui qui est produit avec une méthode de rendu dans laquelle la tête de vecteur prend 20% de la taille totale du vecteur, (b) est celui qui est produit avec une méthode de rendu dans laquelle la tête de vecteur prend 15% de la taille totale du vecteur.

5.3 Accélération du processus d'entraînement par GPU

Nous avons accéléré l'entraînement du réseau de neurones avec les fonctions de perte de la perpendicularité et de la colinéarité (*en anglais, the perpendicularity loss and the same line loss*) par un algorithme profitant du calcul en parallèle avec des fils d'exécution multiples fourni par GPU. L'algorithme conçu contient les étapes suivantes :

- Couper les entrées en plusieurs groupes d'une taille constante définie. Si les dernières entrées ne forment pas un groupe de cette taille, elles deviennent directement le dernier groupe.
- Chaque groupe des entrées est envoyé vers le GPU pour calculer les sommes de la valeur de perte et du gradient selon le poids actuel du réseau. Chaque fil d'exécution appliquent ces instructions :
  - Calculer la valeur de perte et le gradient pour son entrée correspondante selon la fonction de perte choisie.
  - Chaque fil d'exécution du début d'un groupe d'une taille choisie dans l'espace global des fils d'exécution somme les valeurs calculées des membres du groupe. Le fil d'exécution au début des derniers fils d'exécution qui ne forment pas un groupe

ne fait pas cette tâche.

— Le premier fil d'exécution dans l'espace global somme les valeurs sommées dans chaque fil de début des groupes formés et aussi celles des derniers fils qui restent. Il transmet les sommes finales trouvées pour la valeur de perte et le gradient vers le CPU.

— Les valeurs reçues du GPU sont sommées et puis divisées par le nombre total des entrées en vue d'obtenir la valeur de perte et le gradient pour la fonction de perte choisie considérant le poids actuel du réseau.

Le processus d'exécution de cet algorithme est aussi illustré dans la Figure 5.2. Si seulement

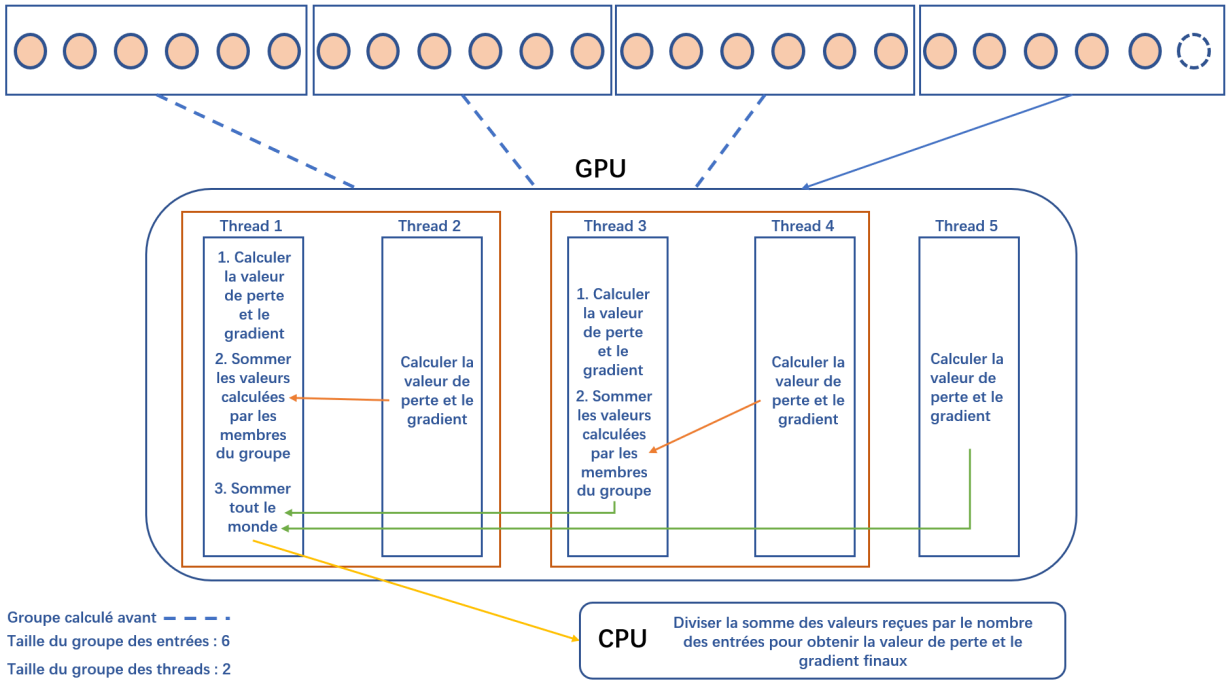


Figure 5.2 Algorithme pour accélérer les calculs de la valeur de perte et du gradient

la fonction de perte de la perpendicularité (*en anglais, the perpendicularity loss*) est utilisée, cet algorithme rend la vitesse d'exécution des calculs de la valeur de perte et du gradient trois fois plus rapide pour les petits champs vectoriels (autour de 500 vecteurs) et sept fois pour les grands champs vectoriels (autour de 50000 vecteurs). Si la fonction de perte de la colinéarité (*en anglais, the same line loss*) est aussi activée, cet algorithme accélère les calculs de la valeur de perte et du gradient par un facteur de 7 pour les petits champs vectoriels (autour de 500 vecteurs). Pour les grands champs vectoriels, la version CPU n'arrive pas à terminer, mais nous pouvons déduire que le facteur d'accélération pourrait être encore plus grand.

5.4 Génération d'animation pour le chemin de caméra

Nous expliquons ici notre méthode pour générer une animation qui montre le parcours sur un chemin de caméra construit. Puisque la façon pour choisir l'orientation de la caméra au cours du chemin a été déjà explicitée dans l'article, nous montrons ici seulement le système de changement de trames utilisé pour l'animation.

Nous avons défini le temps total pour parcourir une section de chemin entre deux points de caméra originalement utilisée dans la construction le chemin de caméra, l'intervalle de temps qui était utilisé pour indiquer après combien de temps la trame devait être changée et aussi l'intervalle de temps limité qui serait utilisé pour la détection du moment du changement de trame. Nous notons séparément ces trois valeurs par T_{total}, T_{change} et T_{limit}.

Le moment de début est enregistré pour l'animation d'une section du chemin de caméra généré. L'algorithme pour animer une section de chemin contient ces étapes :

— Calculer la différence de temps entre le moment actuel et le moment de début enregistré. Cette différence est divisée par T_{total} pour obtenir la valeur t utilisée pour interpoler les informations (la direction de vue et le vecteur up) des deux points de caméra qui sont le départ et la fin de cette section de chemin. Les informations interpolées seront utilisées pour construire le point de caméra correspondant du moment actuel.

— Comparer la différence de temps calculée à T_{total}. Si elle est plus grande que T_{total}, l'algorithme se termine et l'animation est finie.

— Diviser la différence par T_{change} pour trouver le reste.

— Le reste trouvé est comparé à T_{limit}, s'il est inférieur à cet intervalle, la matrice de visualisation sera renouvelée selon le point de vue construit avec les informations trouvées par l'interpolation mentionnée dans la première étape.

Nous bouclons cet algorithme afin d'animer le chemin entre deux points de vue originalement utilisée pour construire le chemin de caméra. Nous appliquons cette méthode pour générer les animations des sections entre toutes les paires consécutives de points de vue sur le chemin généré. Ces sous-animations forment l'animation finale du chemin de caméra au complet.

CHAPITRE 6 CONCLUSION

6.1 Synthèse des travaux

Nous avons développé une nouvelle méthode totalement différente de celles de la série « entropie » qui sélectionne des points de vue de bonne qualité pour des données vectorielles en entraînant un perceptron par les fonctions de perte définies et avons aussi proposé une façon simple pour construire un chemin de caméra en connectant les points de vue choisis avec des splines cubiques.

La méthode proposée pour la sélection de points de vue peut générer des bons points de vue en étant beaucoup plus rapide que l'évaluation d'un bon point de vue selon l'entropie de point de vue. La méthode construite pour la génération du chemin de caméra fournit des résultats acceptables.

6.2 Limitations de la solution proposée

Le choix des hyperparamètres comme le taux d'apprentissage et le nombre d'époques reste manuel. Ce point montre que notre méthode n'est pas entièrement automatique.

6.3 Améliorations futures

Nous pourrions ajouter une étape de la sélection des vecteurs représentatifs dans la méthode afin que le point de vue soit affiché plus correctement.

Le choix du taux d'apprentissage peut être automatisé par un algorithme qui teste des taux d'apprentissage des différents ordres de grandeur et qui garde celui qui commence premièrement à entraîner le réseau. Le choix du nombre d'époques peut devenir non nécessaire si la fin d'entraînement est déterminée par la petite variation des différences de la valeur de perte.

Le domaine d'application de notre méthode de sélection de point de vue peut s'élargir si la squelette du modèle 3D ou de la scène 3D sous la forme de vecteurs peut être construite automatiquement. Ce point est probablement réalisable avec l'algorithme de l'axe médian dans [11].

RÉFÉRENCES

[1] T.-Y. Lee, O. Mishchenko, H.-W. Shen et R. Crawfis, “View point evaluation and streamline filtering for flow visualization,” dans *2011 IEEE Pacific Visualization Symposium*. IEEE, 2011, p. 83–90. [En ligne]. Disponible : https://ieeexplore.ieee.org/document/5742376

[2] P.-P. Vázquez, M. Feixas, M. Sbert et A. Llobet, “Viewpoint entropy : A new tool for obtaining good views of molecules,” dans *Eurographics / IEEE VGTC Symposium on Visualization*. The Eurographics Association, 2002, p. 183–188. [En ligne]. Disponible : http://dx.doi.org/10.2312/VisSym/VisSym02/183-188

[3] K. Muehler, M. Neugebauer, C. Tietjen et B. Preim, “Viewpoint selection for intervention planning,” dans *Eurographics/ IEEE-VGTC Symposium on Visualization*, K. Museth, T. Moeller et A. Ynnerman, édit. The Eurographics Association, 2007, p. 267–274. [En ligne]. Disponible : http://dx.doi.org/10.2312/VisSym/EuroVis07/267-274

[4] E. Monclús, P.-P. Vázquez et I. Navazo, “Efficient selection of representative views and navigation paths for volume data exploration,” dans *Visualization in Medicine and Life Sciences II*, L. Linsen, H. Hagen, B. Hamann et H.-C. Hege, édit. Springer, Berlin, Heidelberg, 2012, p. 133–151. [En ligne]. Disponible : https://doi.org/10.1007/978-3-642-21608-4

[5] J. Tao, J. Ma, C. Wang et C.-K. Shene, “A unified approach to streamline selection and viewpoint selection for 3d flow visualization,” *IEEE Transactions on Visualization and Computer Graphics*, vol. 19, n°. 3, p. 393–406, feb 2013. [En ligne]. Disponible : https://ieeexplore.ieee.org/document/6226391

[6] M. Schelling, P. Hermosilla, P.-P. Vázquez Alcocer et T. Ropinski, “Enabling viewpoint learning through dynamic label generation,” *Computer Graphics Forum*, vol. 40, n°. 2, p. 413–423, may 2021. [En ligne]. Disponible : https://doi.org/10.1111/cgf.142643

[7] P.-P. Vázquez, M. Feixas, M. Sbert et W. Heidrich, “Automatic view selection using viewpoint entropy and its application to image-based modeling,” *Computer Graphics Forum*, vol. 22, p. 689–700, dec 2003. [En ligne]. Disponible : https://doi.org/10.1111/j.1467-8659.2003.00717.x

[8] I. Goodfellow, Y. Bengio et A. Courville, *Deep Learning*. MIT Press, 2016. [En ligne]. Disponible : http://www.deeplearningbook.org

[9] J. Duchi, E. Hazan et Y. Singer, “Adaptive subgradient methods for online learning and stochastic optimization,” *J. Mach. Learn. Res.*, vol. 12, p. 2121–2159, jul 2011. [En ligne]. Disponible : https://dl.acm.org/doi/10.5555/1953048.2021068

[10] D. P. Kingma et J. Ba, “Adam : A method for stochastic optimization,” 2014. [En ligne]. Disponible : https://arxiv.org/abs/1412.6980

[11] W.-H. Hsu, Y. Zhang et K.-L. Ma, “A multi-criteria approach to camera motion design for volume data animation,” *IEEE Transactions on Visualization and Computer Graphics*, vol. 19, n°. 12, p. 2792–2801, dec 2013. [En ligne]. Disponible : https://ieeexplore.ieee.org/abstract/document/6634149

[12] A. Benyoub, “Automatisation de la création de scénarios pour les scènes de la visualisation scientifique,” mémoire de maîtrise, Dép. de génie informatique et génie logiciel, École Polytechnique de Montréal, Montréal, QC, 2015. [En ligne]. Disponible : https://publications.polymtl.ca/1751/

[13] S. Beckhaus, F. Ritter et T. Strothotte, “Cubicalpath-dynamic potential fields for guided exploration in virtual environments,” dans *Proceedings the Eighth Pacific Conference on Computer Graphics and Applications*. IEEE, 2000, p. 387–459. [En ligne]. Disponible : https://ieeexplore.ieee.org/document/883963

[14] A. Amamra, Y. Amara, R. Benaissa et B. Merabti, “Optimal camera path planning for 3d visualisation,” dans *2016 SAI Computing Conference (SAI)*. IEEE, 2016, p. 388–393. [En ligne]. Disponible : https://ieeexplore.ieee.org/abstract/document/7556011

ANNEXE A FIGURES POUR LES TESTS AUXILIAIRES

Nous montrons ici les résultats des tests auxiliaires réalisés pour les autres ensembles de données selon le deuxième groupe de tests mentionné dans l'article du chapitre 4.

Ces résultats vérifient les descriptions sur la fonctionnalité mentionnée dans l'article pour le cas d'utilisation avec toutes les trois fonctions de perte activées.

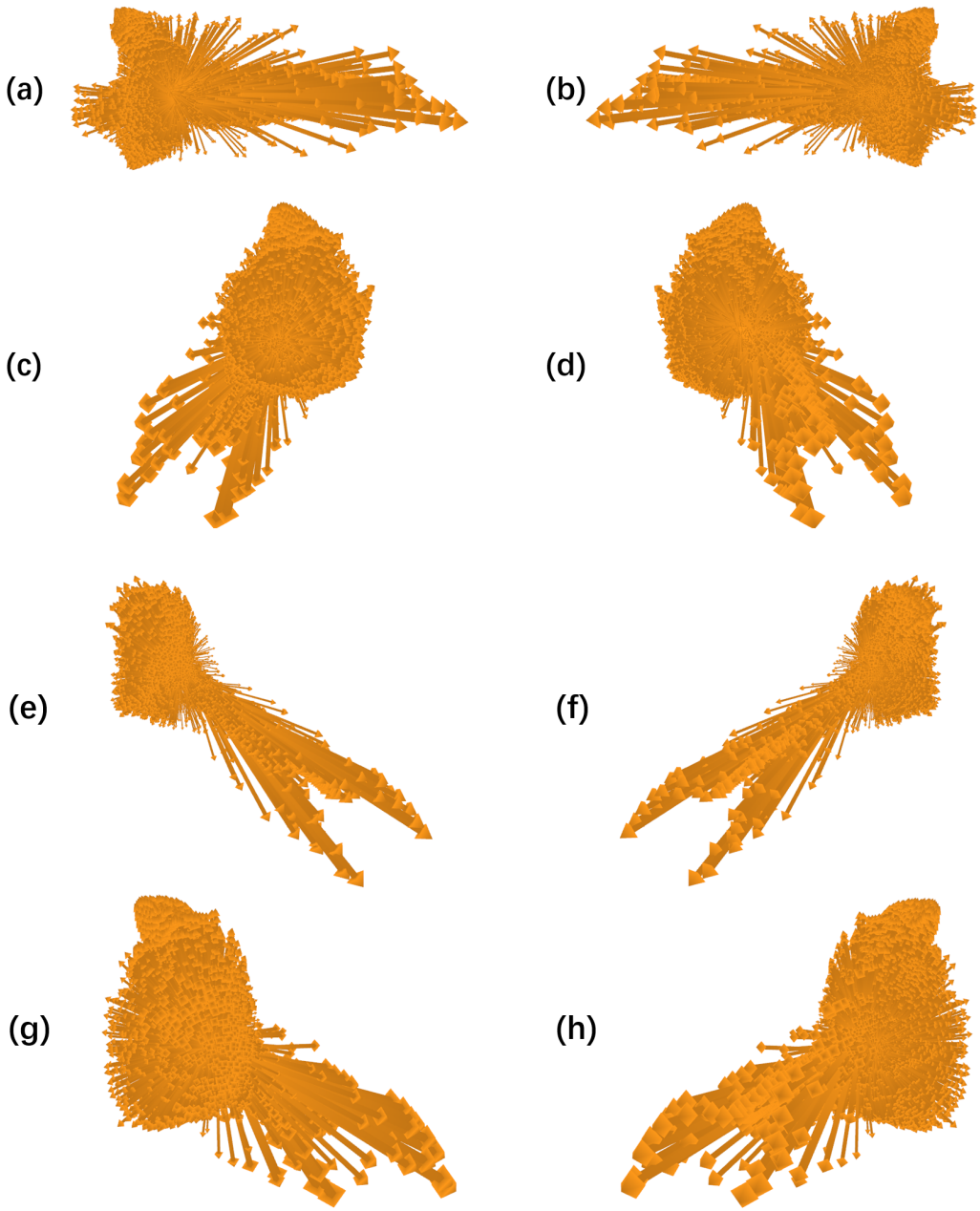


Figure A.1 Points de vue générés pour le modèle « Train » dans le deuxième groupe de tests. Le nombre de vecteurs dans ce modèle est 12071. La fonction de perte de la colinéarité (en anglais, *the same line loss*) utilisée dans ce modèle est la version aléatoire, le nombre des vecteurs aléatoirement pris est 1000. (a)(b) sont les points de vue générés par la fonction de perte de la perpendicularité (en anglais, *the perpendicularity loss*). (c-d-e-f-g-h) sont les points de vue générés avec toutes les trois fonctions de perte. (b-d-f-h) sont les points de vue obtenus par les directions inverses de (a-c-e-g) selon la propriété de la fonction paire. Tous ces points de vue sont obtenus avec un taux d'apprentissage de 0.0005. Le nombre d'époques est 800 pour (a)(b) et 200 pour (c-d-e-f-g-h).

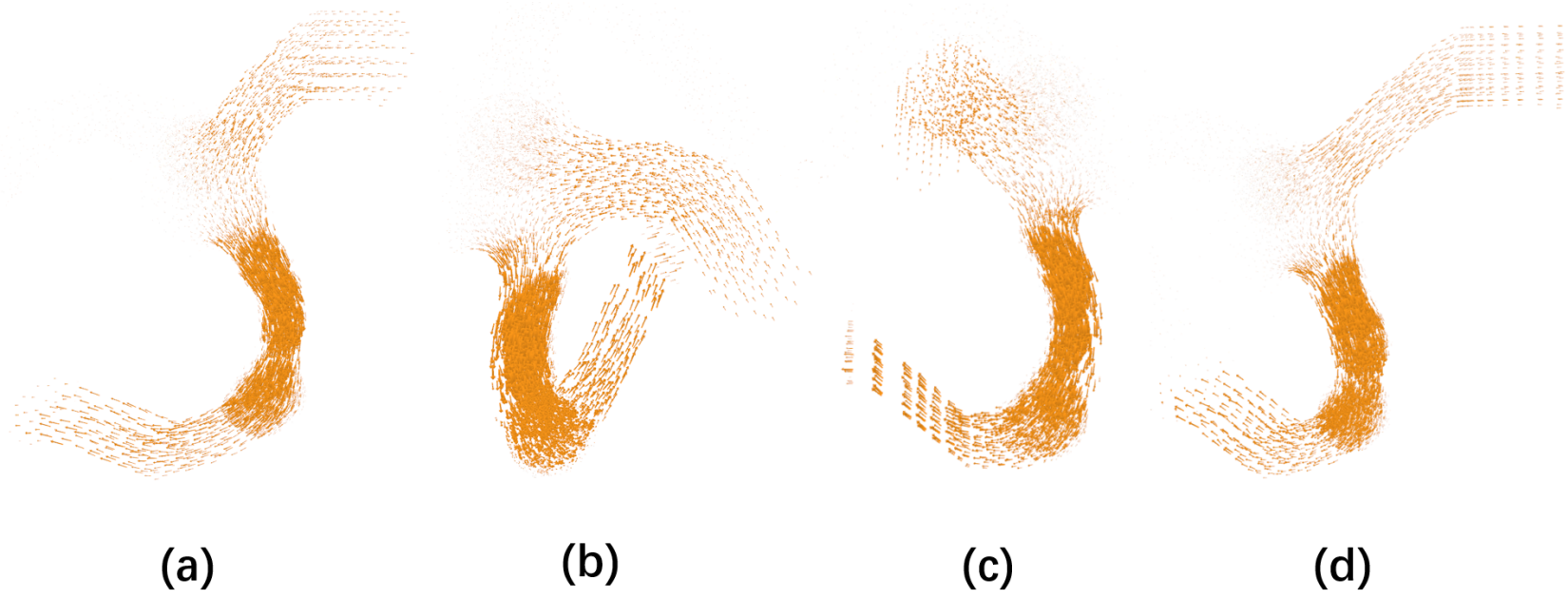


Figure A.2 Points de vue générés pour le modèle « Aneurysm » dans le deuxième groupe de tests. Le nombre de vecteurs dans ce modèle est 23288. La fonction de perte de la colinéarité (en anglais, *the same line loss*) utilisée dans ce modèle est la version aléatoire, le nombre des vecteurs aléatoirement pris est 1000. (a) est le point de vue généré par la fonction de perte de la perpendicularité (en anglais, *the perpendicularity loss*). (b-c-d) sont les points de vue générés avec toutes les trois fonctions de perte. Tous ces points de vue sont obtenus avec un taux d'apprentissage de 1. Le nombre d'époques est 200 pour (a) et 50 pour (b-c-d).

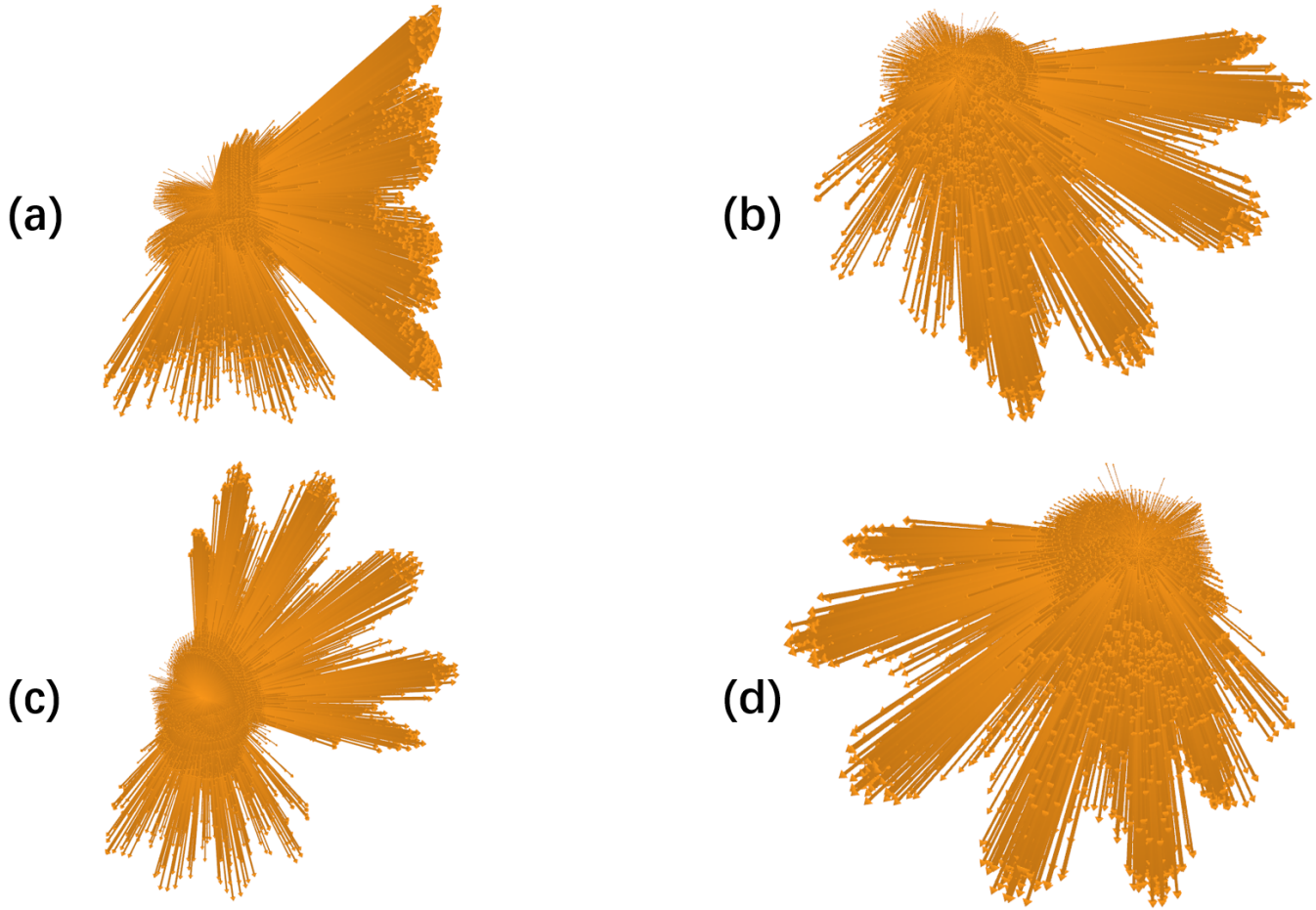


Figure A.3 Points de vue générés pour le modèle « Burner » dans le deuxième groupe de tests. Le nombre de vecteurs dans ce modèle est 49929. La fonction de perte de la colinéarité (en anglais, *the same line loss*) utilisée dans ce modèle est la version aléatoire, le nombre des vecteurs aléatoirement pris est 1000. (a) est le point de vue généré par la fonction de perte de la perpendicularité (en anglais, *the perpendicularity loss*). (b-c-d) sont les points de vue générés avec toutes les trois fonctions de perte. Tous ces points de vue sont obtenus avec un taux d'apprentissage de 0.00005. Le nombre d'époques est 200 pour (a) et 100 pour (b-c-d).

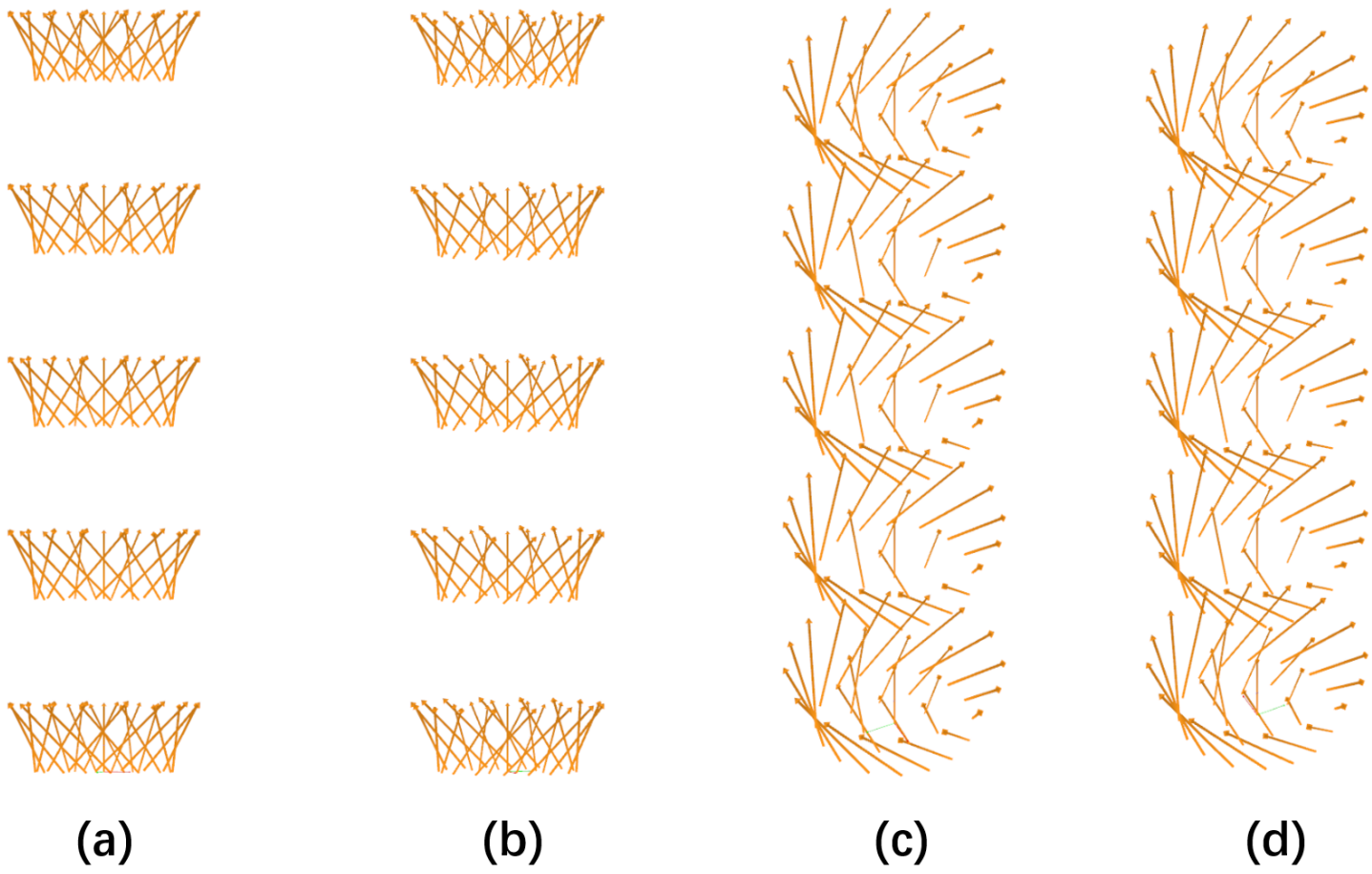


Figure A.4 Points de vue générés pour le modèle « Cylinder » dans le deuxième groupe de tests. Le nombre de vecteurs dans ce modèle est 221. La fonction de perte de la colinéarité (en anglais, *the same line loss*) utilisée dans ce modèle est la version complète. (a) est le point de vue généré par la fonction de perte de la perpendicularité (en anglais, *the perpendicularity loss*). (b-c-d) sont les points de vue générés avec toutes les trois fonctions de perte. Tous ces points de vue sont obtenus avec un taux d'apprentissage de 0.05. Le nombre d'époques est 700 pour (a) et 600 pour (b-c-d).