OpenCCO : Une implémentation de l'algorithme CCO pour la construction d'arbres vasculaires 2D et 3D

Bertrand Kerautret, Phuc Ngo, Nicolas Passat, Hugues Talbot, Clara Jaquet, Adam Germain

OpenCCO : Une implémentation de l'algorithme CCO pour la construction d'arbres vasculaires 2D et 3D

Bertrand Kerautret[1], Phuc Ngo[2], Nicolas Passat[3], Hugues Talbot[4], Clara Jaquet[5]
avec **Adam Germain**[1]

Atelier plénière de cloture de R-Vessel-X 2023
Bron, 15 juin 2023

[1] Université Lyon 2, LIRIS
[2] Université de Lorraine, LORIA
[3] Université de Reims Champagne Ardenne, CReSTIC
[4] CentraleSupelec, Inria
[5] FindOut Diagnostic AB, Sweden

anr

R·Vessel·X
ROBUST VASCULAR NETWORK EXTRACTION
AND UNDERSTANDING WITHIN HEPATIC BIOMEDICAL IMAGES

LIRIS

Plan

1. Introduction

1. Introduction : contexte et motivation initiale (1)

Contexte lié au du projet R-VESSEL-X

- Début de la thèse de Jonas Lamy sur l'analyse de réseau vasculaire du foie (2019).
- Problématique de rehaussement de vaisseaux sur des images IRM.
- Manque de données IRM de vaisseaux annotées sur des images de foie.

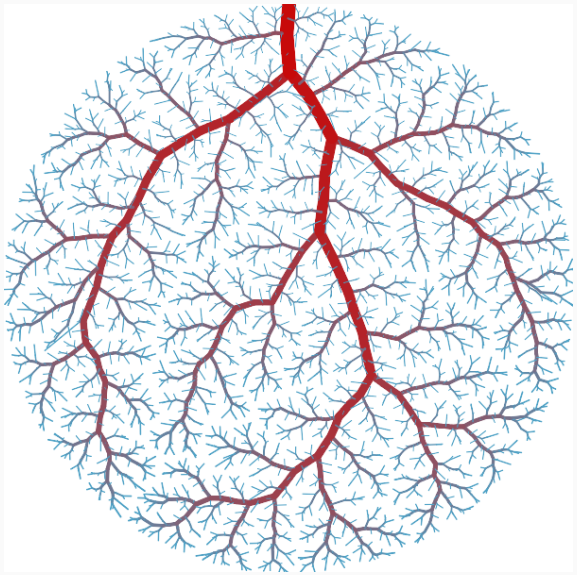
1. Introduction : contexte et motivation initiale (1)

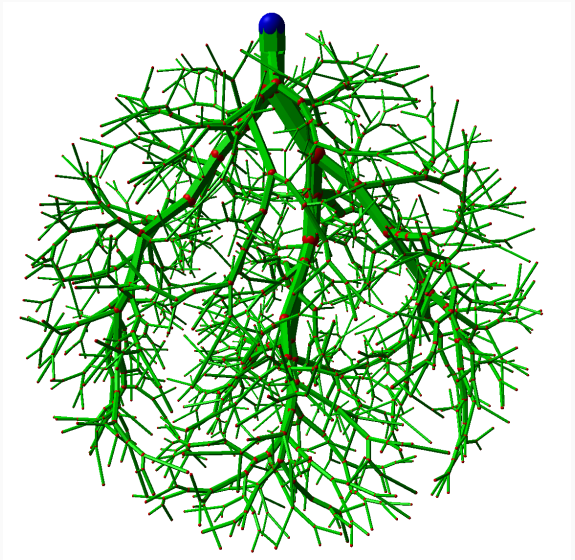
Contexte lié au du projet R-VESSEL-X

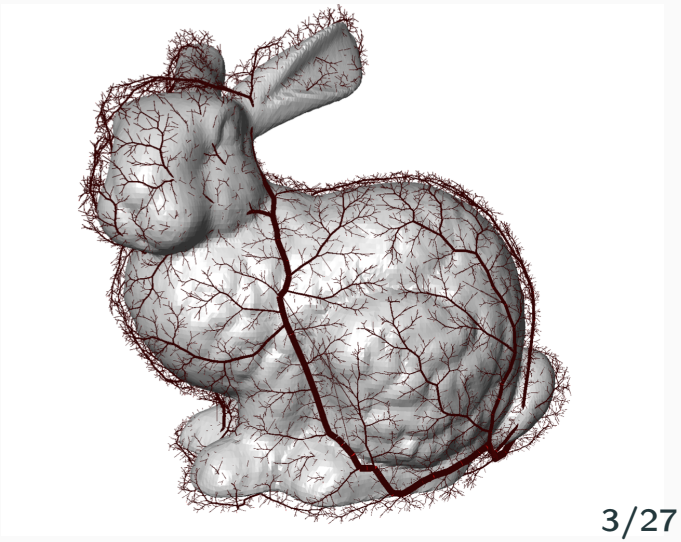
- Début de la thèse de Jonas Lamy sur l'analyse de réseau vasculaire du foie (2019).
- Problématique de rehaussement de vaisseaux sur des images IRM.
- Manque de données IRM de vaisseaux annotées sur des images de foie.

Algorithme CCO : Constrained Constructive Optimization

Créer des objets vasculaires (2D ou 3D) synthétiques mais réalistes d'un point vue anatomique.



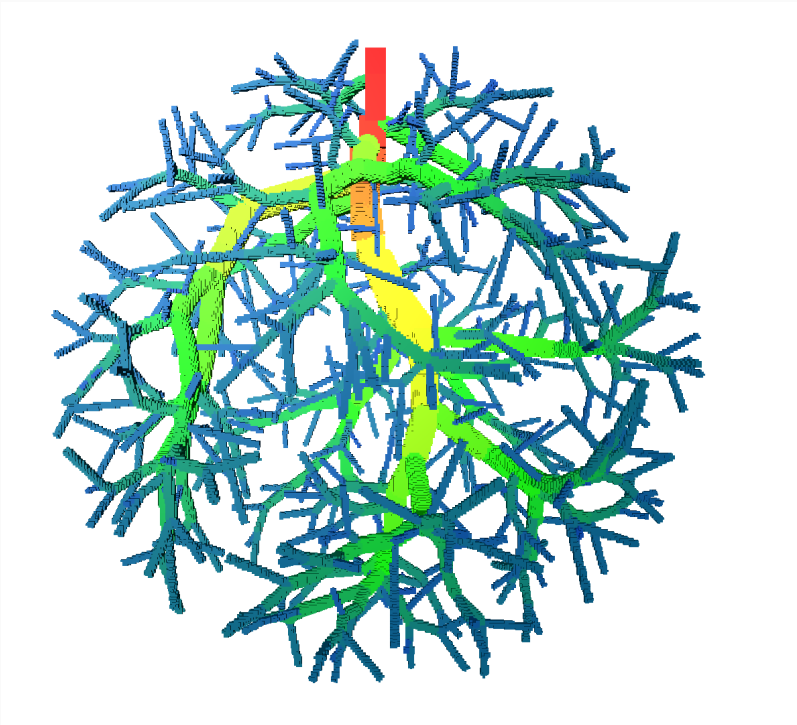




1. Introduction : contexte et motivation initiale (2)

Nombreux points d'intérêt

- Génération d'images médicales de synthèse.
- Apprentissage (à travers les méthodes DL).
- Vérité terrain pour benchmarks (filtrage, analyse géométrique).
- Utilisation des modèles pour la simulation/modélisation (perfusion/vasculaire).



1. Introduction : contexte et motivation initiale (2)

Nombreux points d'intérêt

- Génération d'images médicales de synthèse.
- Apprentissage (à travers les méthodes DL).
- Vérité terrain pour benchmarks (filtrage, analyse géométrique).
- Utilisation des modèles pour la simulation/modélisation (perfusion/vasculaire).

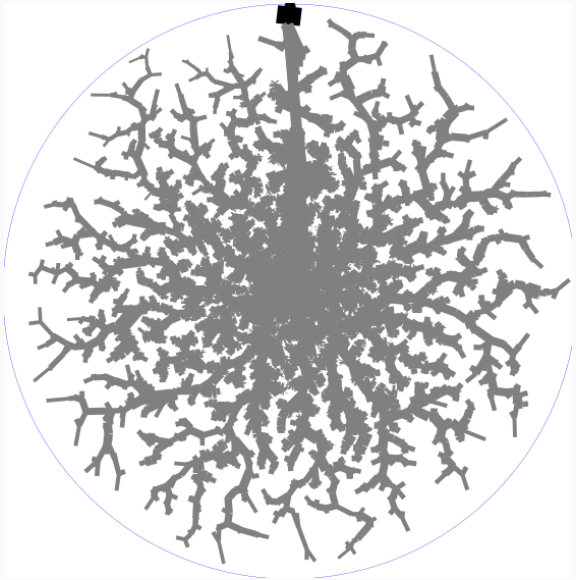
Existant

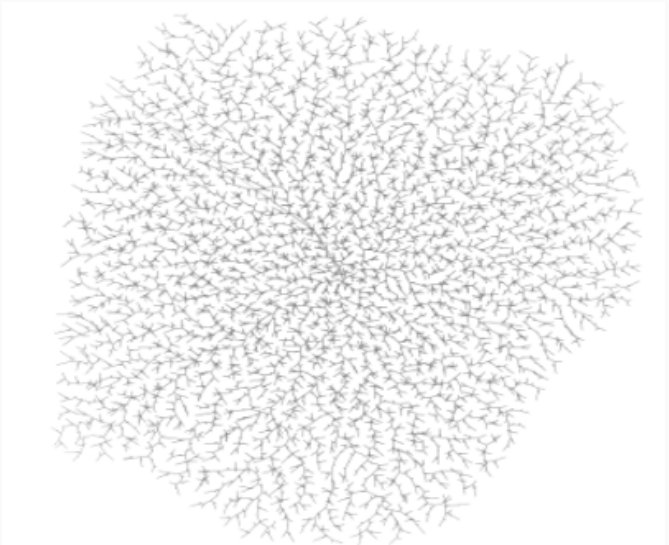
- Schreiner et al. Computer-optimization of vascular trees. IEEE TBE 1993
- VascuSynth : domaines cubiques, uniquement en 3D, optimisation simplifiée.
- Thèse de Clara Jaquet (Hugues Talbot, Paris Est, 2018)
  - Jaquet et al. Generation of Patient-Specific Cardiac Vascular Networks: A Hybrid Image-Based and Synthetic Geometric Model. IEEE TBE 2019
  - Collaboration industrielle avec HeartFlow : code non disponible

1. Introduction : principales étapes de travail

Principales étapes:

- 1. Stage de Master 2 : Dylan Framery mars-sept 2020
  $\Rightarrow$ base de l'algorithme, construction de l'arbre sans optimisation.

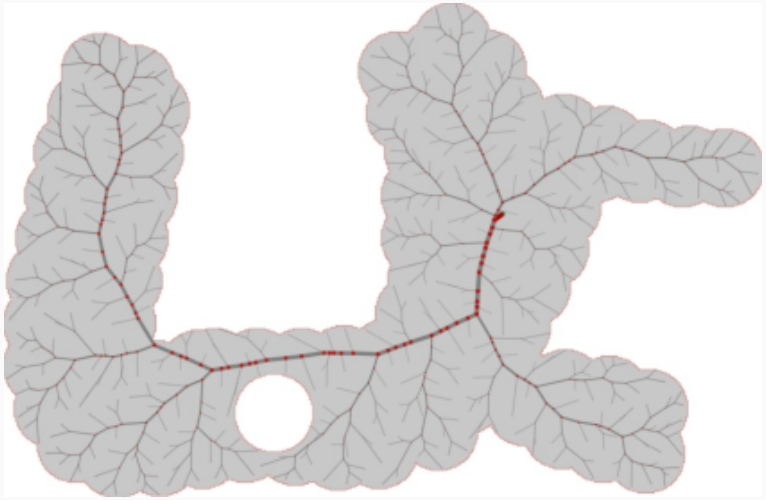




1. Introduction : principales étapes de travail

Principales étapes:

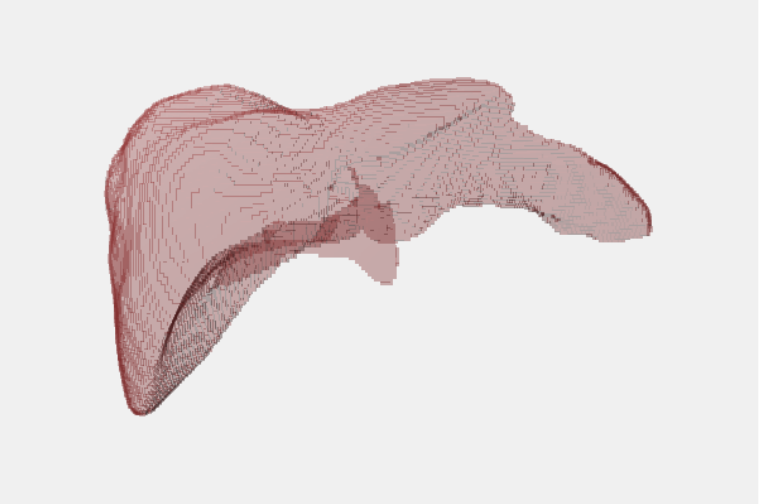
- 1. Stage de Master 2 : Dylan Framery mars-sept 2020
  $\Rightarrow$ base de l'algorithme, construction de l'arbre sans optimisation.
- 2. Reprise du code et algorithme à Roscoff avec Phuc et Nicolas juin 2021.



1. Introduction : principales étapes de travail

Principales étapes:

- 1. Stage de Master 2 : Dylan Framery mars-sept 2020
  $\Rightarrow$ base de l'algorithme, construction de l'arbre sans optimisation.
- 2. Reprise du code et algorithme à Roscoff avec Phuc et Nicolas juin 2021.
- 3. Finalisation, contrôle du domaine, gestion 2D/3D et travail soumis à IPOL.



1. Introduction : principales étapes de travail

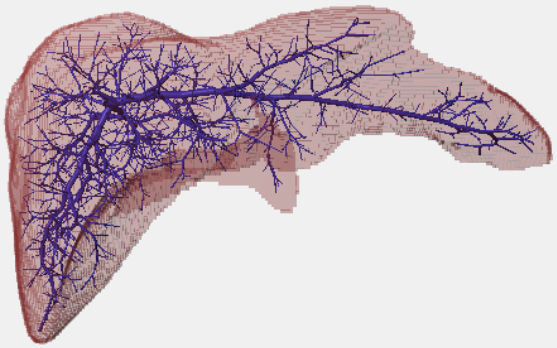
Principales étapes:

- 1. Stage de Master 2 : Dylan Framery mars-sept 2020
  $\Rightarrow$ base de l'algorithme, construction de l'arbre sans optimisation.
- 2. Reprise du code et algorithme à Roscoff avec Phuc et Nicolas juin 2021.
- 3. Finalisation, contrôle du domaine, gestion 2D/3D et travail soumis à IPOL.

1. Introduction : principales étapes de travail

Principales étapes:

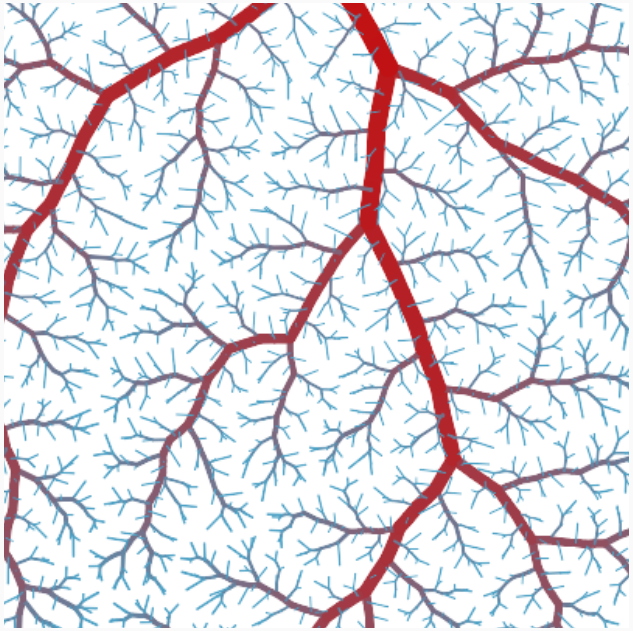
- 1. Stage de Master 2 : Dylan Framery mars-sept 2020
  $\Rightarrow$ base de l'algorithme, construction de l'arbre sans optimisation.
- 2. Reprise du code et algorithme à Roscoff avec Phuc et Nicolas juin 2021.
- 3. Finalisation, contrôle du domaine, gestion 2D/3D et travail soumis à IPOL.



1. Introduction : principales étapes de travail

Principales étapes:

- 1. Stage de Master 2 : Dylan Framery mars-sept 2020
  $\Rightarrow$ base de l'algorithme, construction de l'arbre sans optimisation.
- 2. Reprise du code et algorithme à Roscoff avec Phuc et Nicolas juin 2021.
- 3. Finalisation, contrôle du domaine, gestion 2D/3D et travail soumis à IPOL.
- 4. Stage de Master 1 : Adam travail sur le rendu d'images basé CCO (depuis mai).
  $\rightarrow$ co encadré avec O. Merveille (CREATIS, projet collaboration DIGITBIOMED).

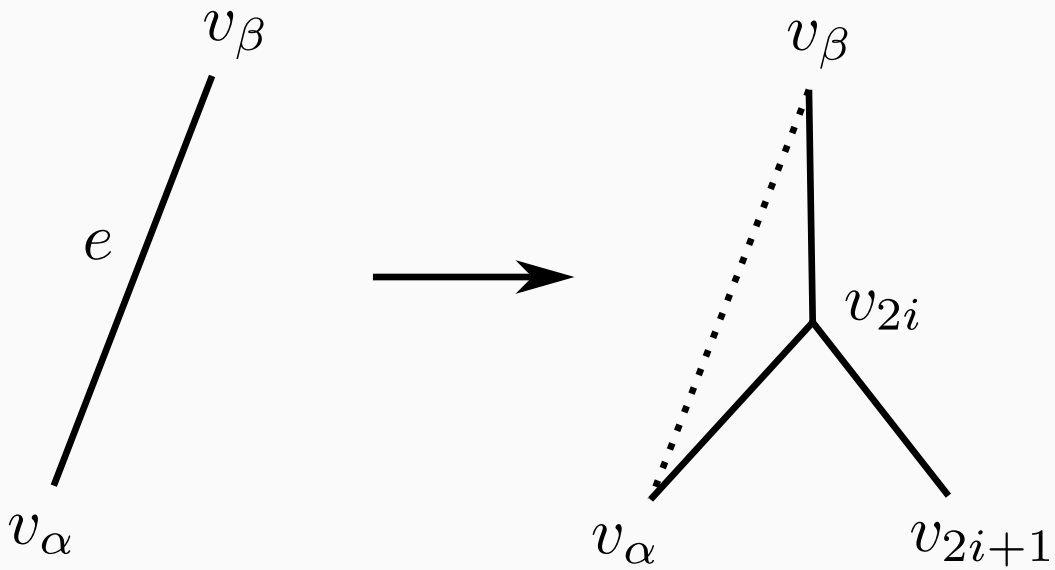


2. Rappel points clés de l'algorithme

Idée générale de l'algorithme

Construction incrémentale en n étapes identiques

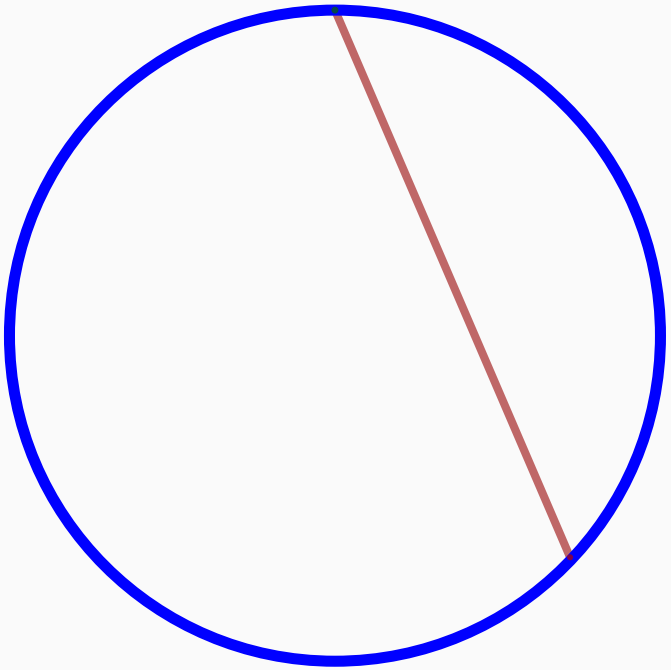
- Ajout d'un terminal + une branche incidente (= une bifurcation)



Idée générale de l'algorithme

Construction incrémentale en n étapes identiques

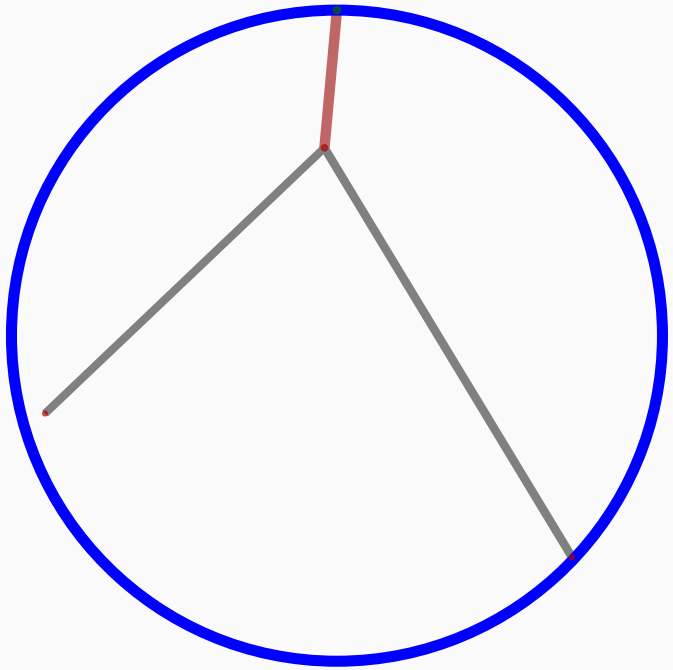
- Ajout d'un terminal + une branche incidente (= une bifurcation)
- Pour plusieurs candidats $\Rightarrow$ **Choix du meilleur candidat**
  - Sélection d'une branche pour connexion
  - Test d'éligibilité (intersection, proximité, ...)
  - **Calcul du point de bifurcation**
  - (Re)calcul des diamètres des branches ascendantes.



Idée générale de l'algorithme

Construction incrémentale en n étapes identiques

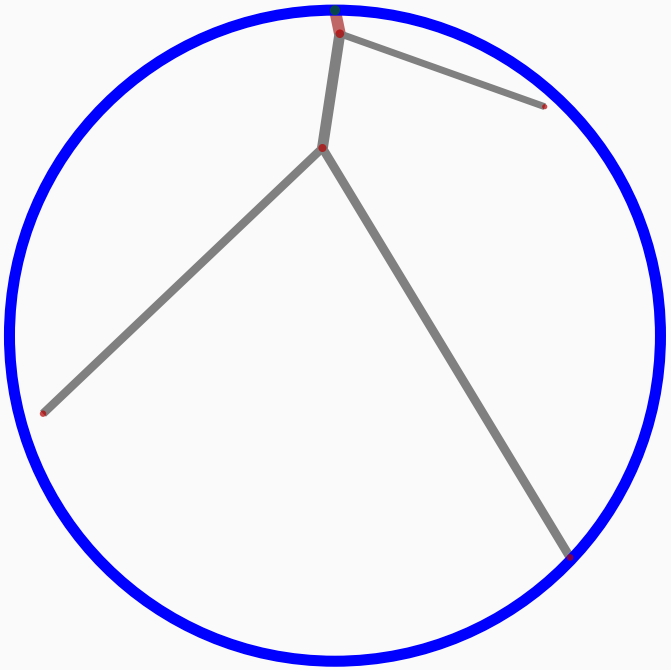
- Ajout d'un terminal + une branche incidente (= une bifurcation)
- Pour plusieurs candidats $\Rightarrow$ **Choix du meilleur candidat**
  - Sélection d'une branche pour connexion
  - Test d'éligibilité (intersection, proximité, ...)
  - **Calcul du point de bifurcation**
  - (Re)calcul des diamètres des branches ascendantes.



Idée générale de l'algorithme

Construction incrémentale en n étapes identiques

- Ajout d'un terminal + une branche incidente (= une bifurcation)
- Pour plusieurs candidats $\Rightarrow$ **Choix du meilleur candidat**
  - Sélection d'une branche pour connexion
  - Test d'éligibilité (intersection, proximité, ...)
  - **Calcul du point de bifurcation**
  - (Re)calcul des diamètres des branches ascendantes.



Idée générale de l'algorithme

Construction incrémentale en n étapes identiques

- Ajout d'un terminal + une branche incidente (= une bifurcation)
- Pour plusieurs candidats $\Rightarrow$ **Choix du meilleur candidat**
  - Sélection d'une branche pour connexion
  - Test d'éligibilité (intersection, proximité, ...)
  - **Calcul du point de bifurcation**
  - (Re)calcul des diamètres des branches ascendantes.

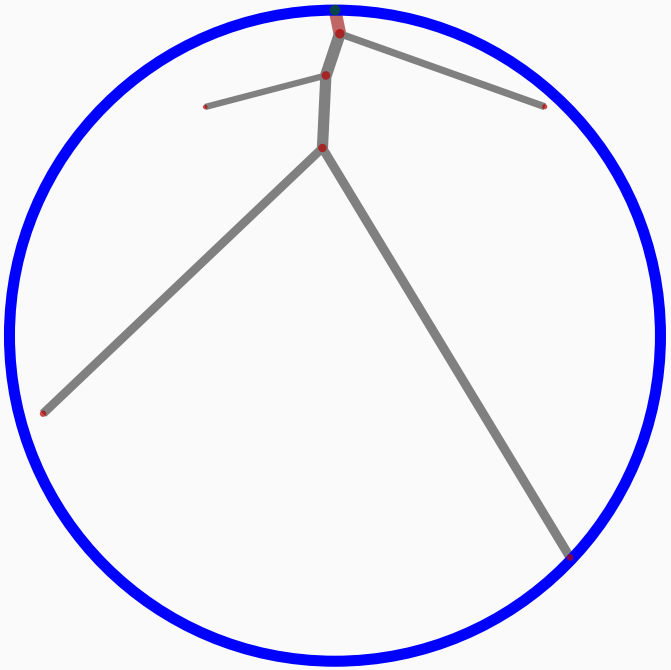


Schéma algorithmique

Algorithme (aspects topologiques)

Input: Nombre de points terminaux $N_{term} \in \mathbb{N}_+^*$

Result: **Graphe** de l'arbre vasculaire $(\mathcal{V}, \mathcal{E})$

`// Initialisation du graphe`

1 Let v_0, v_1 be two vertices of the first segment.

2 Let $(\mathcal{V}_0, \mathcal{E}_0) = (\{v_0, v_1\}, \{(v_0, v_1)\})$.

`// Ajout des segments au graphe`

3 **foreach** $i \in [\![1, N_{term} - 1]\!]$ **do**

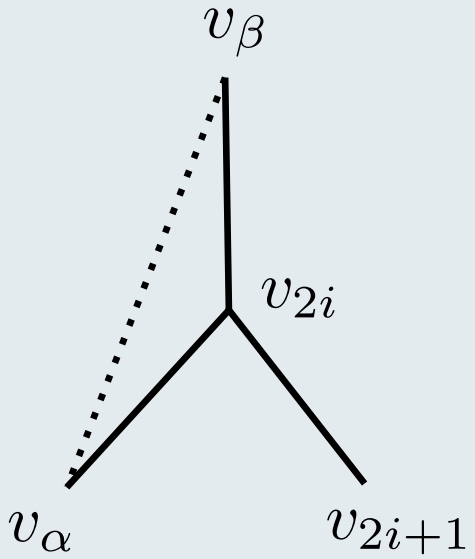
4 Let v_{2i}, v_{2i+1} be two vertices.

5 Choisir $e = (v_\alpha, v_\beta) \in \mathcal{E}_{i-1}$ (avec $0 \leq \alpha, \beta < i$)

6 Let $\mathcal{V}_i = \mathcal{V}_{i-1} \cup \{v_{2i}, v_{2i+1}\}$

7 Let $\mathcal{E}_i = (\mathcal{E}_{i-1} \setminus \{e\}) \cup \{(v_\alpha, v_{2i}), (v_{2i}, v_\beta), (v_{2i}, v_{2i+1})\}$

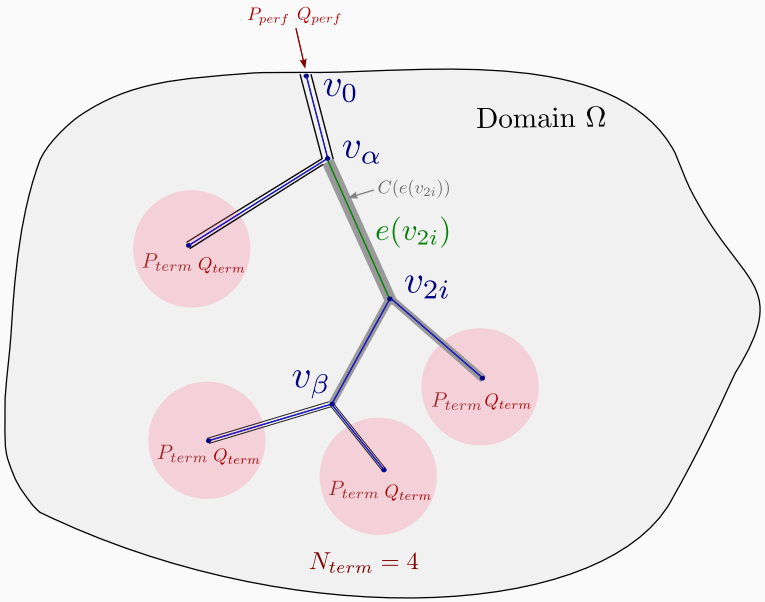
8 Let $(\mathcal{V}, \mathcal{E}) = (\mathcal{V}_{N_{term}-1}, \mathcal{E}_{N_{term}-1})$



Paramètres

Paramètres algorithmiques

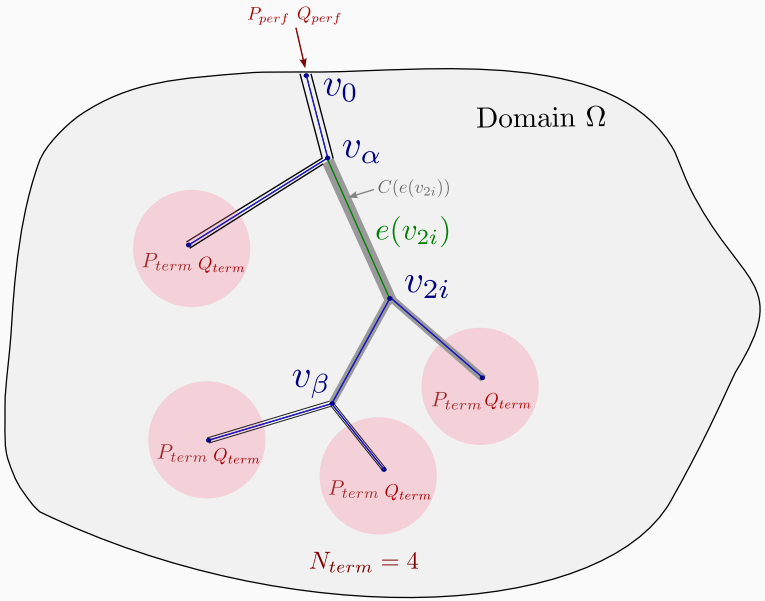
- Dimension $d \in \{2, 3\}$
- Domaine $\Omega \subset \mathbb{R}^d$ (convexe ou non convexe) et la racine $x_0 \in \Omega$
- Nombre de points terminaux $N_{term} \in \mathbb{N}^*$
- Nombre des plus proches segments $N_{neigh} \in \mathbb{N}^* \Rightarrow$ test de connexion
- Nombre des plus proches segments $N_{max} \in \mathbb{N}^* \Rightarrow$ test d'intersection



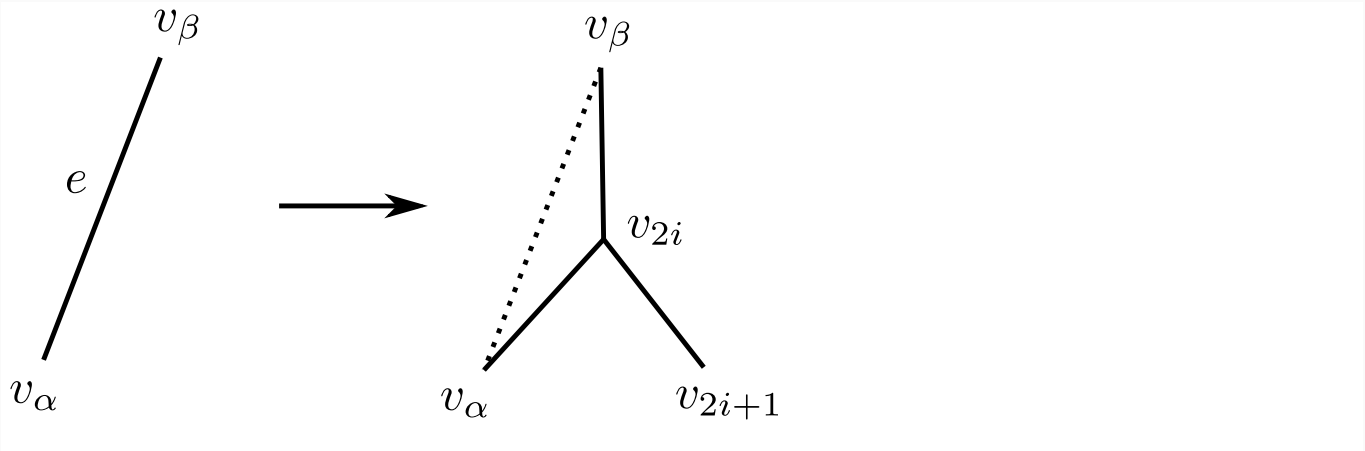
Paramètres

Paramètres physiologiques

- Volume de perfusion (A_{perf})
- Pression d'entrée, de sortie (P_{perf} et P_{term})
- Flot d'entrée, de sortie (Q_{perf} et Q_{term})
- Viscosité (μ)
- Coefficient de bifurcation de Murray (γ)



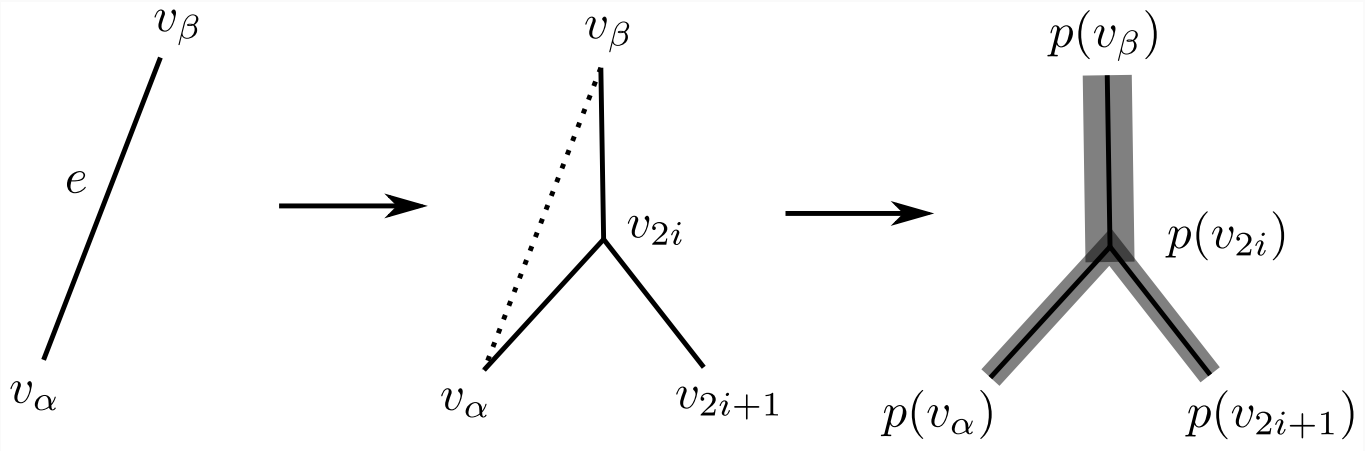
Construction topologique mais aussi géométrique...



Aspects géométriques liés au plongement

- Chaque sommet a un plongement dans $\mathbb{R}^d$
- Chaque arête est associée à un segment de droite entre deux sommets

Construction topologique mais aussi géométrique...



Aspects géométriques liés au plongement

- Chaque sommet a un plongement dans $\mathbb{R}^d$
- Chaque arête est associée à un segment de droite entre deux sommets

Contraintes liées au plongement :

- Chaque sommet et segment doivent être complètement dans Ω
- Aucun segment ne doit "s'intersecter" ($\rightarrow$ épaisseur)
- Les sommets doivent être suffisamment éloignés de l'arbre
- Chaque segment doit être suffisamment long

... et les contraintes physiologiques

Contraintes physiologiques

- mesure du domaine (aire en 2D et volume en 3D)
- pression (entrée ; sorties)
- flot (entrée ; sorties)
- viscosité
- coefficient de bifurcation

... et les contraintes physiologiques

Contraintes physiologiques

- mesure du domaine (aire en 2D et volume en 3D)
- pression (entrée ; sorties)
- flot (entrée ; sorties)
- viscosité
- coefficient de bifurcation

Pour trouver une solution optimale

- la connexion minisant le **volume total** de l'arbre
- position optimale de la **bifurcation**
- mises à jour des diamètres des branches
- test d'éligibilité : intersection, proximité, ...

Synthèse du processus global

Trois phases distinctes

- Choix de points terminaux aléatoires pour la construction :
  - contraintes : domaine, distance, intersection, ...
  - multiplicité : choix du segment de rattachement
    $\rightarrow$ les N_{neigh} segments les plus proches

Synthèse du processus global

Trois phases distinctes

- Choix de points terminaux aléatoires pour la construction :
  - contraintes : domaine, distance, intersection, ...
  - multiplicité : choix du segment de rattachement
    $\rightarrow$ les N_{neigh} segments les plus proches
- Processus d'optimisation :
  - position optimale du point de la bifurcation
  - contraintes : domaine, longueur, intersection, ...
    $\rightarrow$ les N_{max} segments les plus proches
  - minimisation globale du volume

Synthèse du processus global

Trois phases distinctes

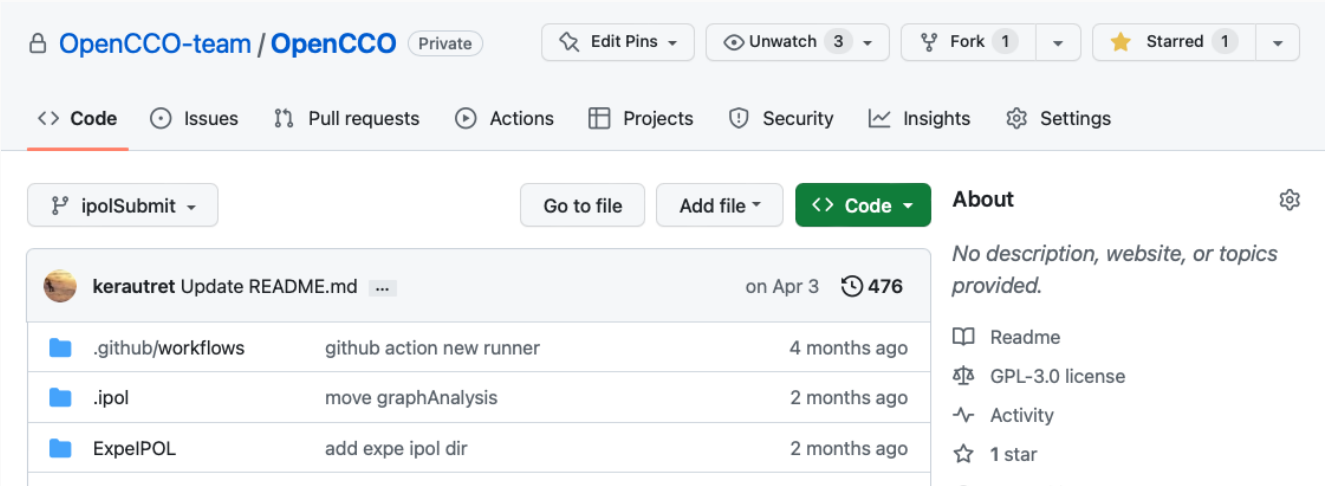
- Choix de points terminaux aléatoires pour la construction :
  - contraintes : domaine, distance, intersection, ...
  - multiplicité : choix du segment de rattachement
    $\rightarrow$ les N_{neigh} segments les plus proches
- Processus d'optimisation :
  - position optimale du point de la bifurcation
  - contraintes : domaine, longueur, intersection, ...
    $\rightarrow$ les N_{max} segments les plus proches
  - minimisation globale du volume
- Mises à jour :
  - rayons des branches ascendantes
  - pressions, flots, volume global, ...

3. Implémentation

3. Implémentation : organisation

Organisation logicielle

- Langage de programmation : `C++`
- Outils de compilation: `cmake`
- Dépendances :
  - Ceres Solver : solveur non linéaire pour l'optimisation géométrique
  - DGtal : gestion de domaine 2D/3D, visualisation
- Hébergement: *GitHub*, outils CI: Linux/MacOs)
- Image *docker*.



3. Implémentation : niveau utilisateur (1)

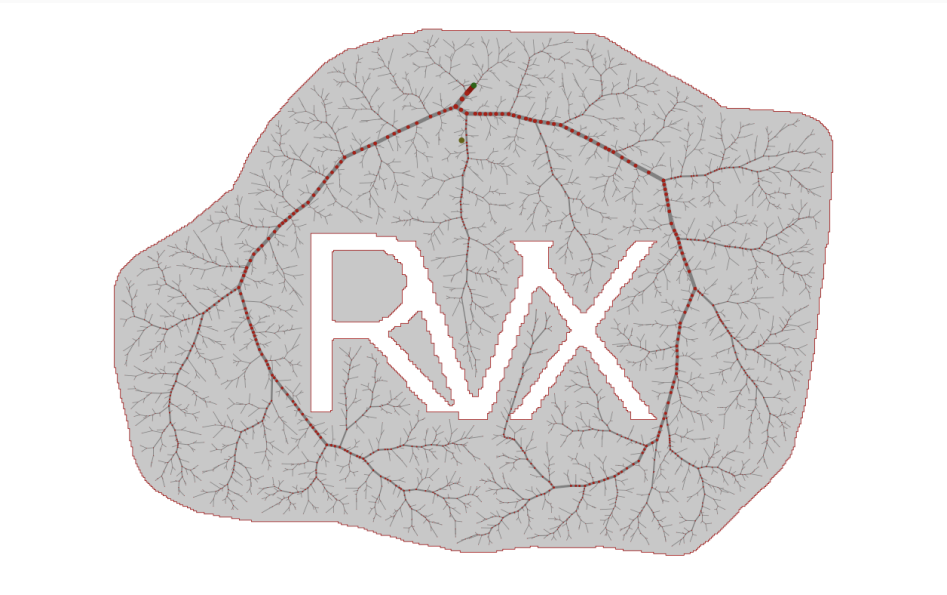
Executables en ligne de commandes

- Orientation générale : pouvoir scripter la génération d'arbre.

3. Implémentation : niveau utilisateur (1)

Executables en ligne de commandes

- Orientation générale : pouvoir scripter la génération d'arbre.
- `bin/generateTree2D` et `generateTree3D` : génération de l'arbre vasculaire 2D/3D
  $\rightarrow$ export de l'arbre en format xml

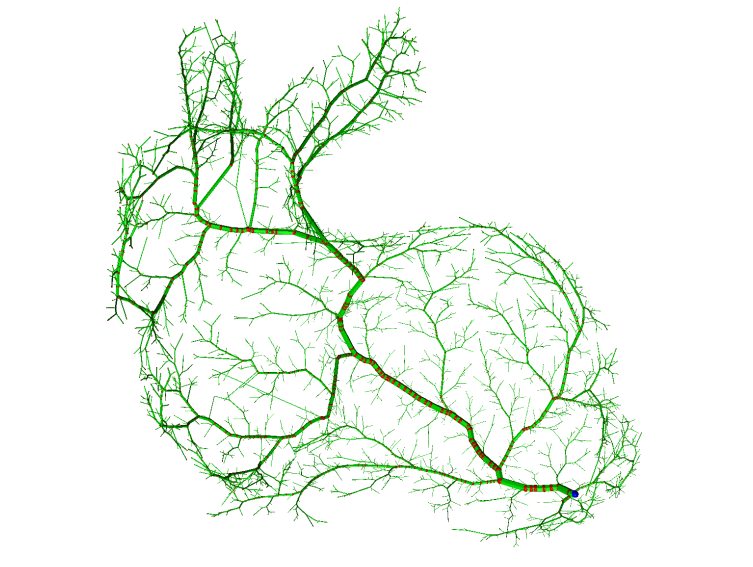


```
./build/bin/generateTree2D -n 3000 -a 20000 -d Samples/shape3.pgm
```

3. Implémentation : niveau utilisateur (1)

Executables en ligne de commandes

- Orientation générale : pouvoir scripter la génération d'arbre.
- `bin/generateTree2D` et `generateTree3D` : génération de l'arbre vasculaire 2D/3D
  $\rightarrow$ export de l'arbre en format xml

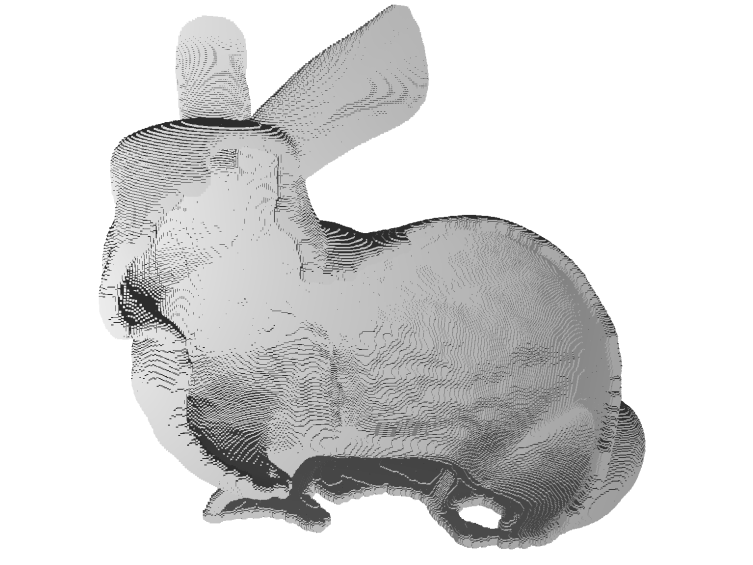


```
generateTree3D -n 3000 -a 20000 -d Samples/bunnyThickBdr.vol -view -m 1
-p 143 -107 7
```

3. Implémentation : niveau utilisateur (1)

Executables en ligne de commandes

- Orientation générale : pouvoir scripter la génération d'arbre.
- `bin/generateTree2D` et `generateTree3D` : génération de l'arbre vasculaire 2D/3D $\rightarrow$ export de l'arbre en format xml

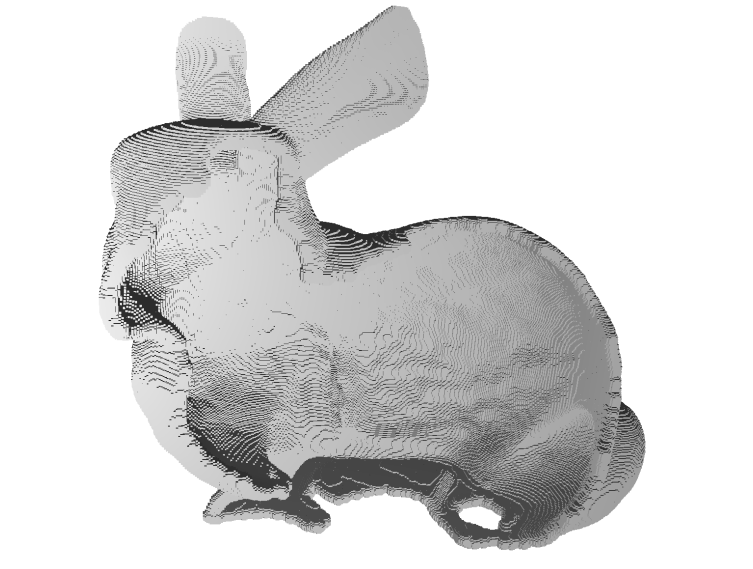


3dVolBoundaryViewer Samples/bunnyThickBdr.vol

3. Implémentation : niveau utilisateur (1)

Executables en ligne de commandes

- Orientation générale : pouvoir scripter la génération d'arbre.
- `bin/generateTree2D` et `generateTree3D` : génération de l'arbre vasculaire 2D/3D
  $\rightarrow$ export de l'arbre en format xml
- `tools/graphAnalysis` : outils d'analyse/conversion depuis format xml.
  $\rightarrow$ utile aussi pour le format xml généré par VascuSynth



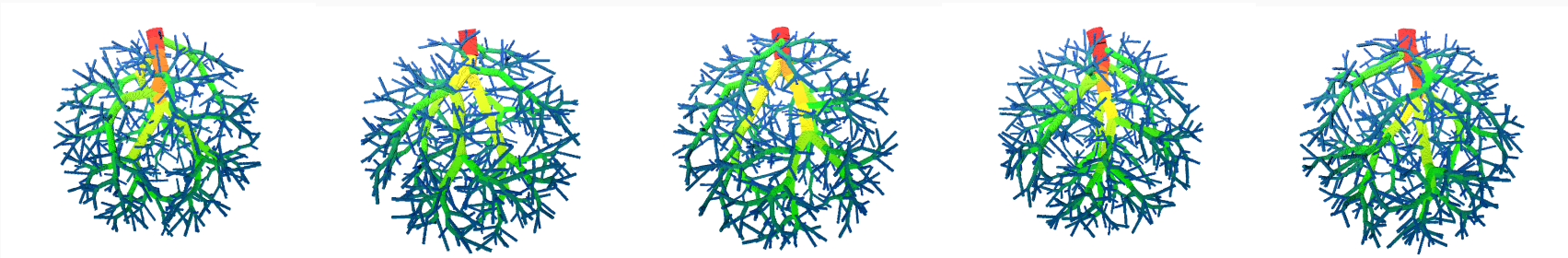
3dVolBoundaryViewer Samples/bunnyThickBdr.vol

3. Implémentation : niveau utilisateur (2)

Outils interfaçables avec les tools de DGtalTools-Conbtrib

Exemple de script pour générer 100 volumes discrétisés :

```
#!/bin/bash
typeset -Z3 i
for i in $(seq -w 1 100)
do
   generateTree3D
   xml2graph tree_3D.xml r
   graph2vol -v r_vertex.dat -e r_edges.dat -r r_radius.dat -o vol${i}.nii -g 10
done
```

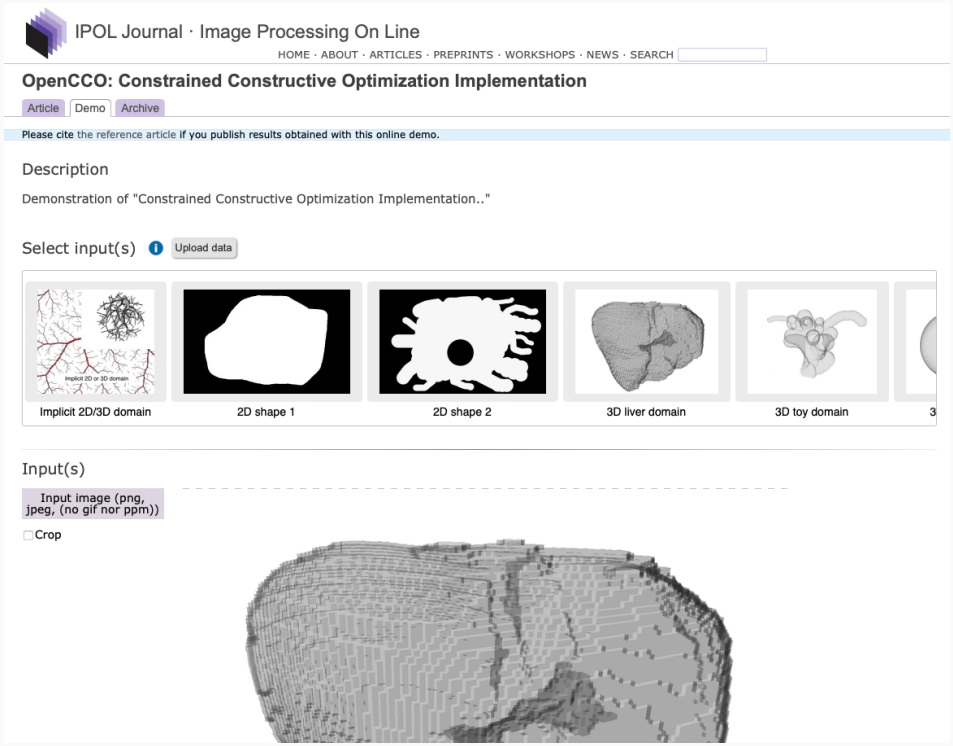


...

3. Implémentation : niveau utilisateur (3)

Démonstration en ligne (associé à la soumission IPOL)

- Définie en 2D et 3D.
- Domaines : implicite, image de masque 3D donné par l'utilisateur.
- Export disponible en ligne, format xml.



3. Implémentation : niveau développeur (1)

Structure du code : deux classes importantes

- **CoronaryArteryTree** : structure de l'arbre vasculaire (paramètres physiologiques, segments et rayons, définition du domaine...).

3. Implémentation : niveau développeur (1)

Structure du code : deux classes importantes

- **CoronaryArteryTree** : structure de l'arbre vasculaire (paramètres physiologiques, segments et rayons, définition du domaine...).
- **DomainController** : contrôle le domaine d'expansion de l'arbre.
  Concept qui demande les fonctions `randomPoint()`, `isInside()`, `checkNoIntersectDomain()`

3. Implémentation : niveau développeur (1)

Structure du code : deux classes importantes

- **CoronaryArteryTree** : structure de l'arbre vasculaire (paramètres physiologiques, segments et rayons, définition du domaine...).
- **DomainController** : contrôle le domaine d'expansion de l'arbre.
  Concept qui demande les fonctions `randomPoint()`, `isInside()`, `checkNoIntersectDomain()`
  → Plusieurs classes suivent le concept du DomainController :
  - `CircularDomainCtrl` : domaine circulaire.
  - `SquareDomainCtrl` : domaine carré.
  - `ImageMaskDomainCtrl` : domaine défini à travers l'image associé à un masque.

3. Implémentation : niveau développeur (1)

Structure du code : deux classes importantes

- **CoronaryArteryTree** : structure de l'arbre vasculaire (paramètres physiologiques, segments et rayons, définition du domaine...).
- **DomainController** : contrôle le domaine d'expansion de l'arbre.
  Concept qui demande les fonctions `randomPoint()`, `isInside()`, `checkNoIntersectDomain()`
  → Plusieurs classes suivent le concept du DomainController :
  - `CircularDomainCtrl` : domaine circulaire.
  - `SquareDomainCtrl` : domaine carré.
  - `ImageMaskDomainCtrl` : domaine défini à travers l'image associé à un masque.

Structure commune 2D et 3D

- Classe `CoronaryArteryTree` : deux paramètres template :
  → dimension du domaine 2D/3D.
  → contrôleur de domaine.
- Contrôleurs de domaine: un paramètre template associé à la dimension.

3. Implémentation : niveau développeur (2)

Exemple de code minimal :

Reconstruction en trois étapes :

1. Définition des types et construction de l'objet arbre :

```
typedef SquareDomainCtrl<2> SqDomCtrl;
typedef  CoronaryArteryTree<SqDomCtrl, 2> TTree;
SqDomCtrl::TPoint pCenter (0,0);
SqDomCtrl aCtr(1.0,pCenter);
TTree aTree  (aPerf, nbTerm, aCtr, 1.0);
```

3. Implémentation : niveau développeur (2)

Exemple de code minimal :

Reconstruction en trois étapes :

1. Définition des types et construction de l'objet arbre :

```
typedef SquareDomainCtrl<2> SqDomCtrl;
typedef  CoronaryArteryTree<SqDomCtrl, 2> TTree;
SqDomCtrl::TPoint pCenter (0,0);
SqDomCtrl aCtr(1.0,pCenter);
TTree aTree  (aPerf, nbTerm, aCtr, 1.0);
```

2. Initialisation et expansion de l'arbre :

```
ExpandTreeHelpers::initFirtElemTree(aTree, verbose);
ExpandTreeHelpers::expandTree(aTree);
```

3. Implémentation : niveau développeur (2)

Exemple de code minimal :

Reconstruction en trois étapes :

1. Définition des types et construction de l'objet arbre :

```
typedef SquareDomainCtrl<2> SqDomCtrl;
typedef  CoronaryArteryTree<SqDomCtrl, 2> TTree;
SqDomCtrl::TPoint pCenter (0,0);
SqDomCtrl aCtr(1.0,pCenter);
TTree aTree  (aPerf, nbTerm, aCtr, 1.0);
```

2. Initialisation et expansion de l'arbre :

```
ExpandTreeHelpers::initFirtElemTree(aTree, verbose);
ExpandTreeHelpers::expandTree(aTree);
```

3. Export du résultat :

```
XMLHelpers::writeTreeToXml(aTree, "tree_2D.xml");
```

3. Implémentation : niveau développeur (3)

Extension simple à effectuer :

Ex : définir une génération d'arbre guidée par les intensités d'une image source.

- Implémenter un nouveau contrôleur de domaine.
- Hériter de `ImageMaskDomainCtrl`.
- Surcharger la méthode `RandomPoint()`.

3. Implémentation : niveau développeur (3)

Extension simple à effectuer :

Ex : définir une génération d'arbre guidée par les intensités d'une image source.

- Implémenter un nouveau contrôleur de domaine.
- Hériter de `ImageMaskDomainCtrl`.
- Surcharger la méthode `RandomPoint()`.

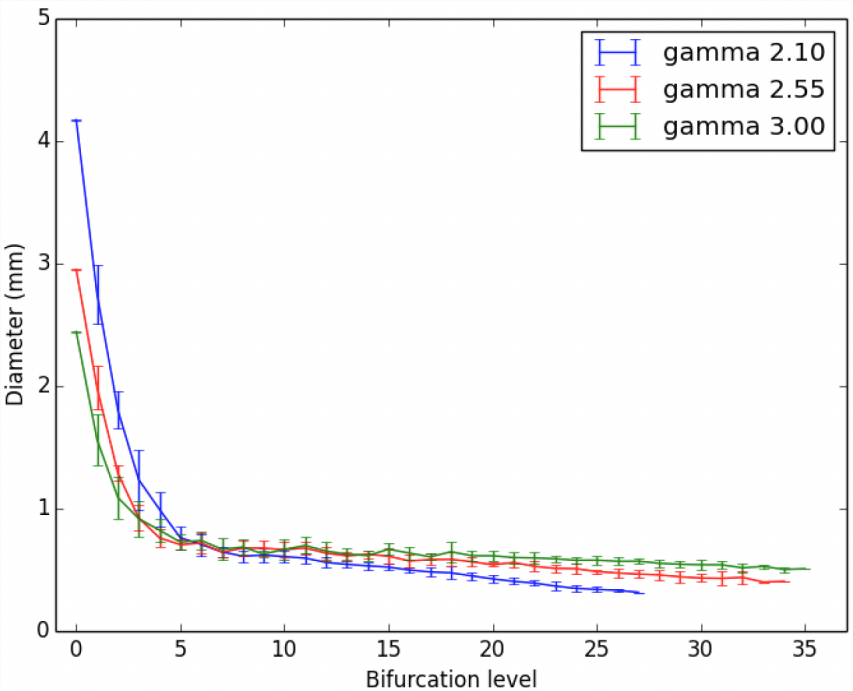
Autres perspectives

Faire évoluer deux arbres vasculaires sur le même domaine.

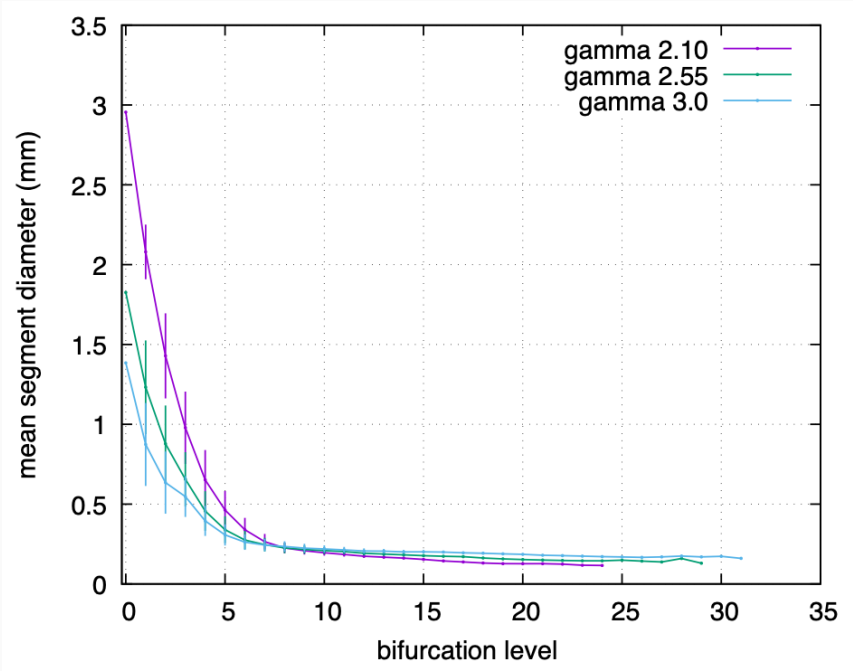
- Contrôleur de domaine avec une référence sur un autre arbre.
- Méthode spécifique `RandomPoint()`, `checkNoIntersectDomain()`.

4. Résultats

Analyse des diamètres moyens



Jaquet



OpenCCO

Temps d'exécution

Paramètres physiologiques

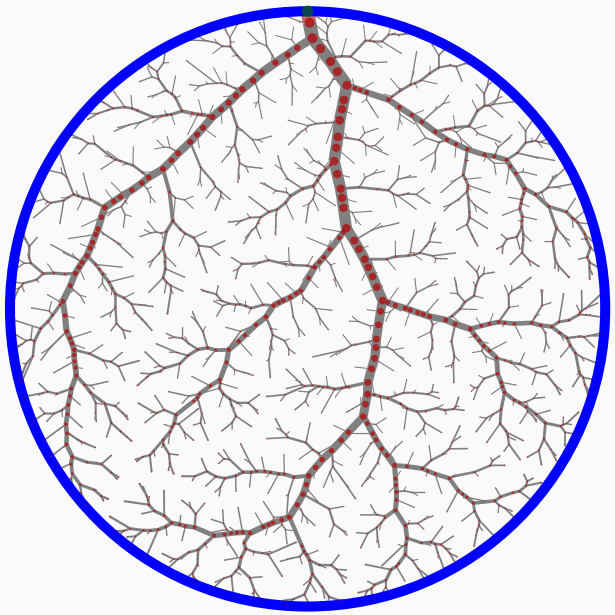
- Perfusion surface / volume : $A_{perf} = 20000\ cm^2\ /\ cm^3$
- Perfusion pressure : $P_{perf} = 1.33e4Pa$
- Terminal pressure : $P_{term} = 8.33e3mm^3/s$
- Viscosity of blood : $\mu = 3.6cp$
- Murray bifurcation exponent : $\gamma = 3$

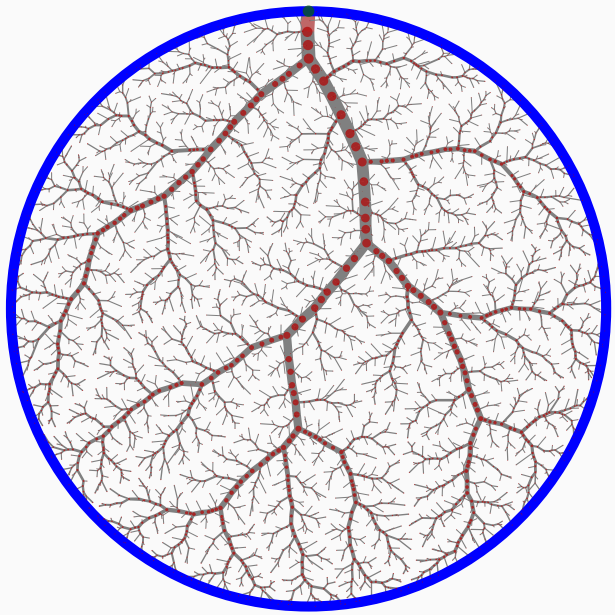
Paramètres d'algorithme

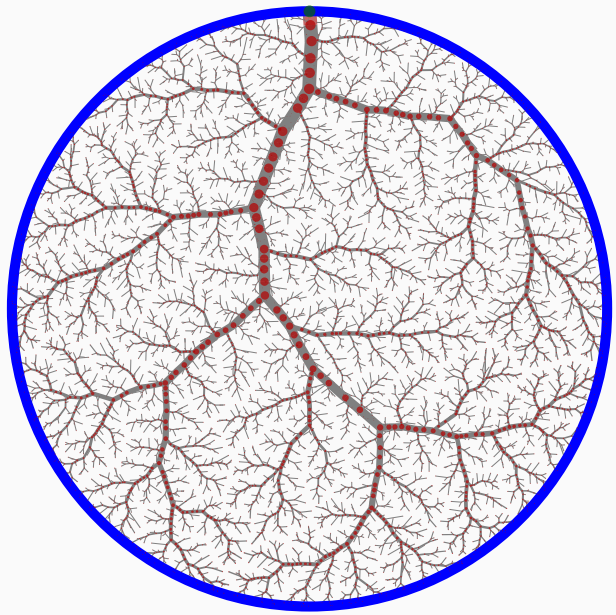
- Nombre de terminaux : $N_{term} \in \mathbb{N}_+^*$
- Nombre de segments pour le test de proximité : $N_{max} = 20$
- Nombre de segments pour le test d'intersection : $N_{neigh} = 20$

Temps d'exécution

Modèle	N_{term}	Temps d'exécution
2D	1000	129 s
	2000	478 s
	3000	1024 s
	4000	1897 s
	5000	3435 s

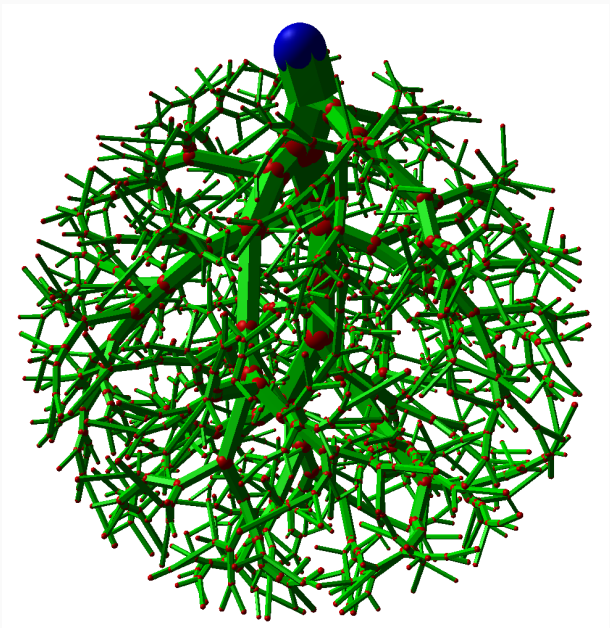


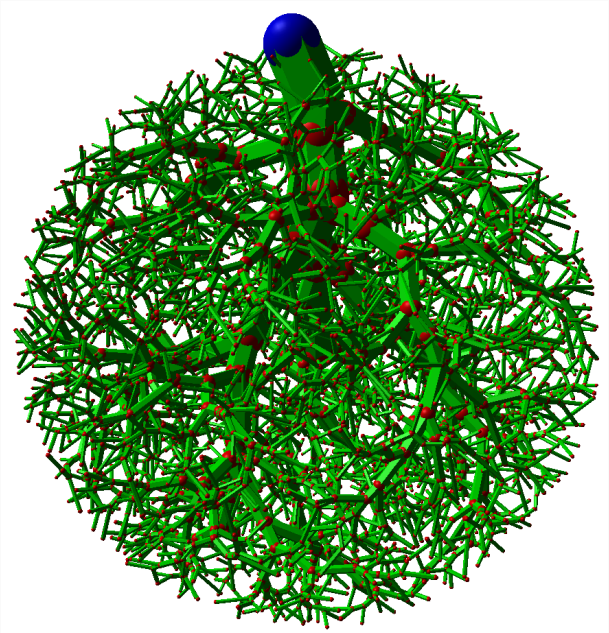


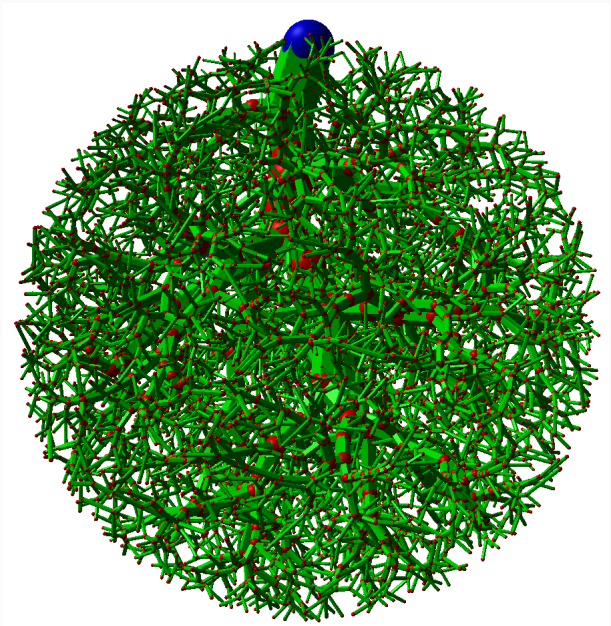


Temps d'exécution

Modèle	N_{term}	Temps d'exécution
3D	1000	219 s
	2000	902 s
	3000	1780 s
	4000	3835 s
	5000	5451 s

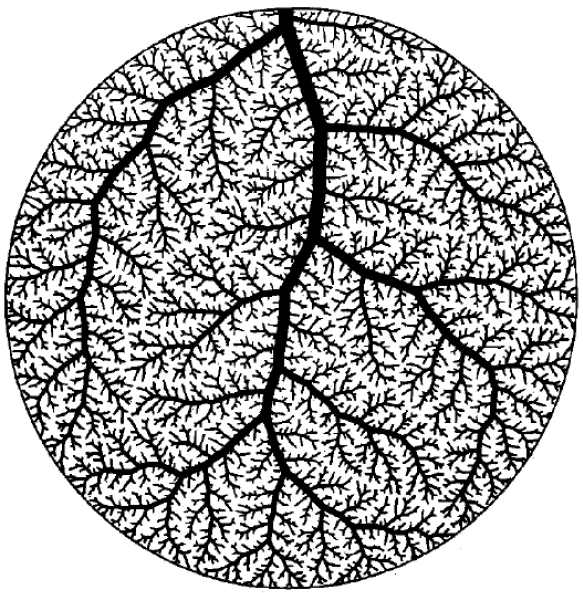


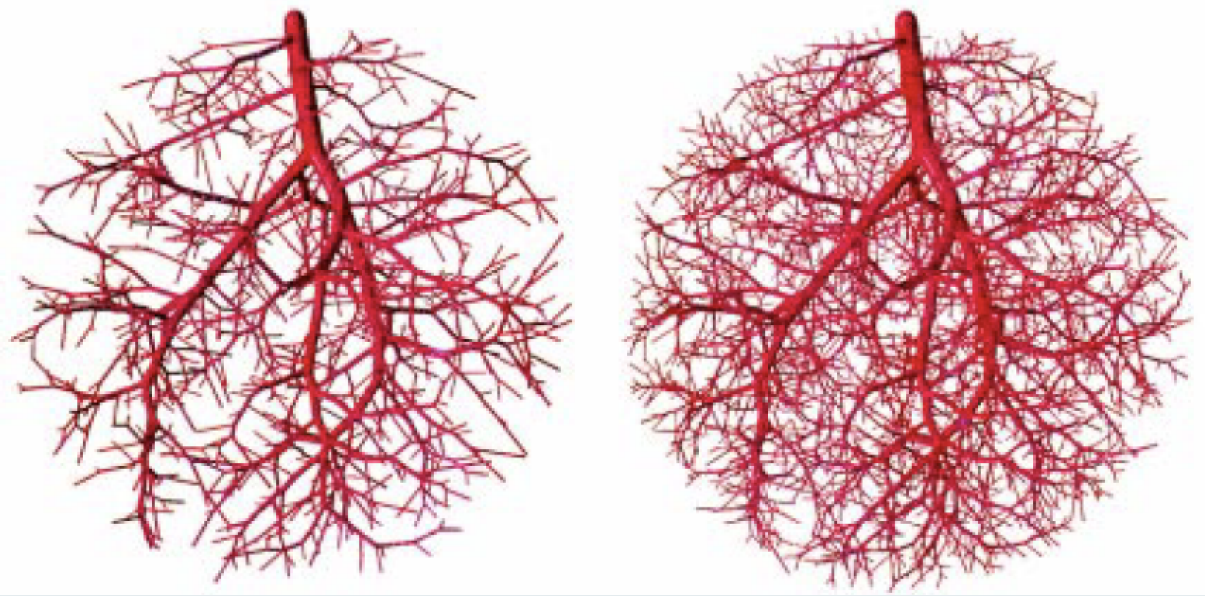




Résultats antérieurs

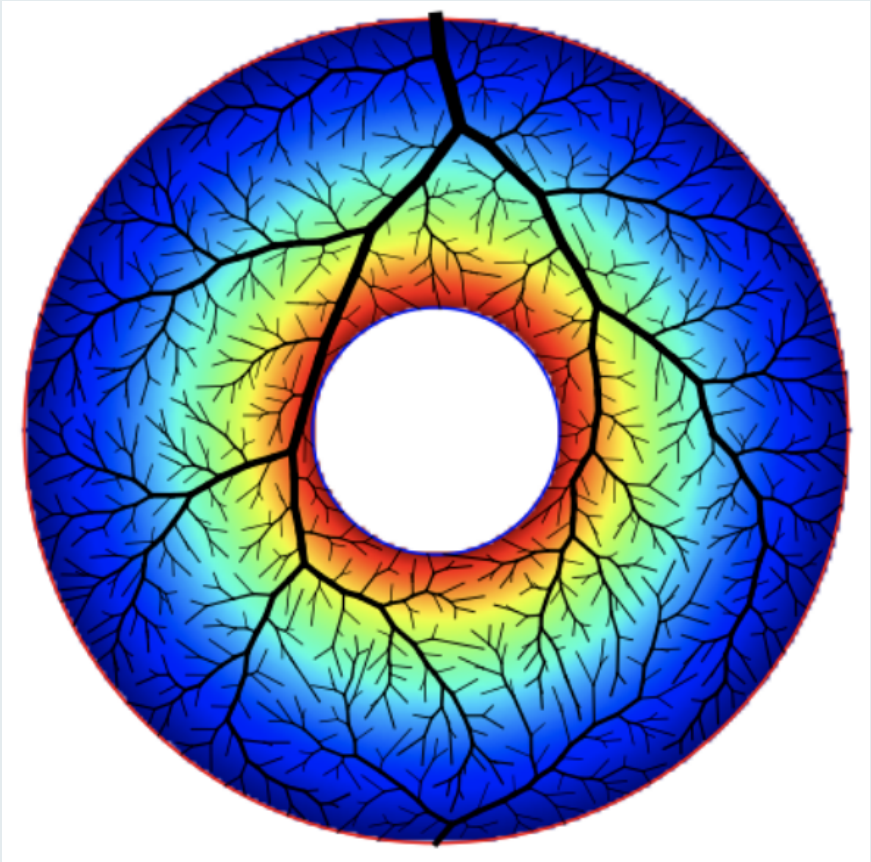
CCO 2D (Schreiner) et 3D (Karch)

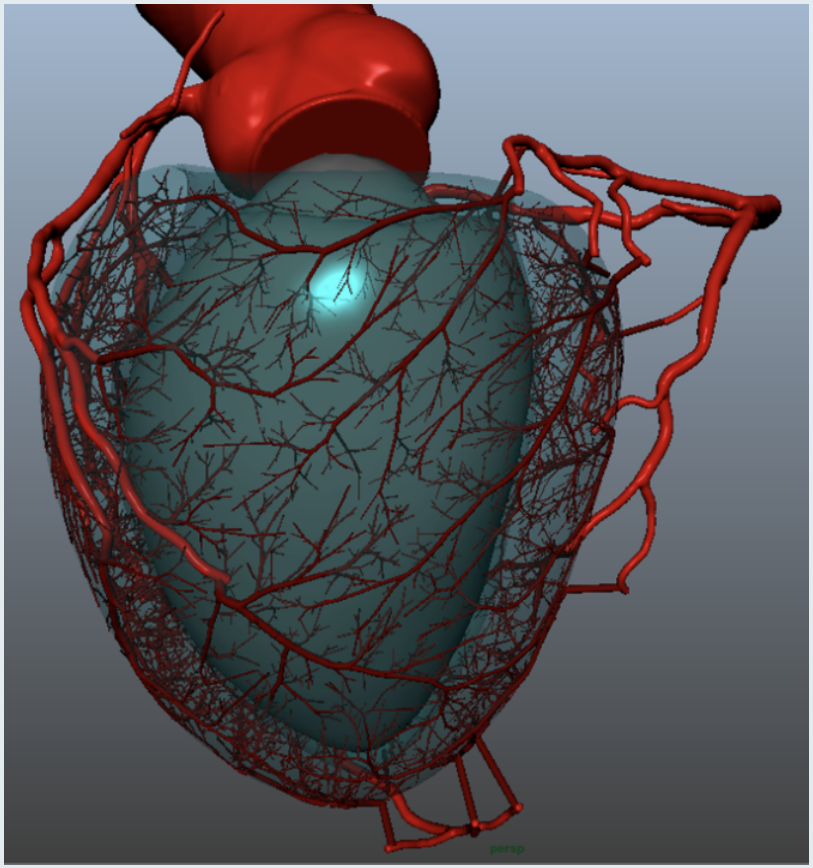




Résultats antérieurs

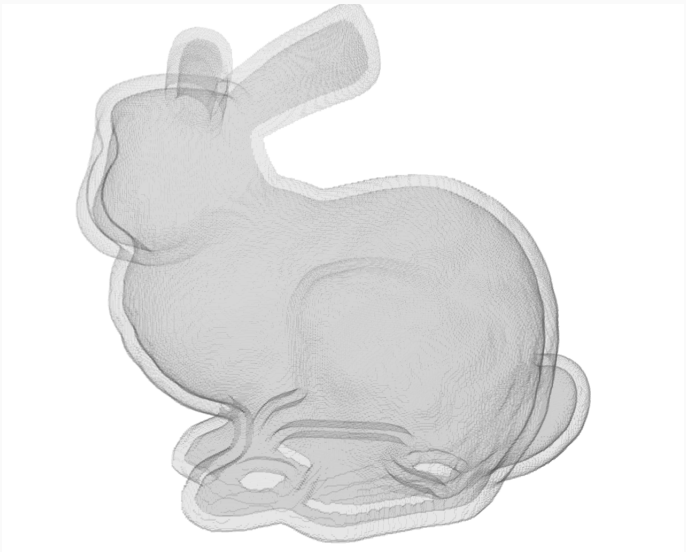
CCO 2D & 3D (Jaquet)

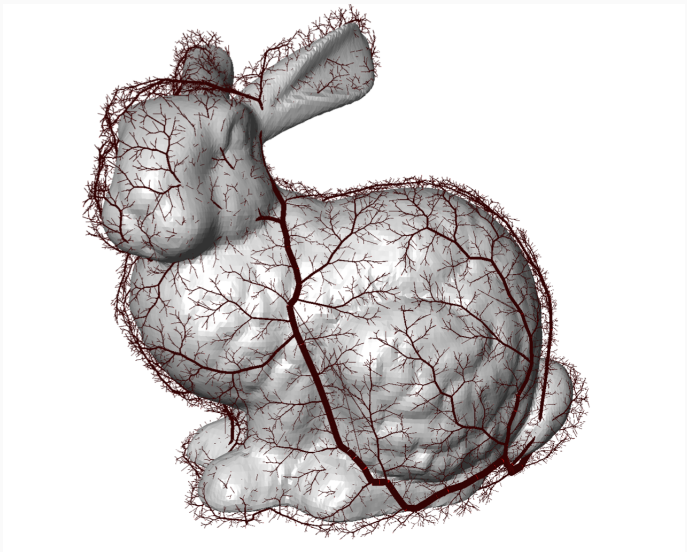




Autres résultats récents

CCO sur domaine non convexe





5. Conclusion

Conclusion

- Implémentation fonctionnelle 2D / 3D en C++.
- Démo en ligne et dépôt GitHub.
- Article IPOL soumis.
- Étude comparative avec VascuSynth.

Perspectives (initiées avec le stage Adam Germain)

- Amélioration du rendu avec les valeurs flux.
- Génération de rendu volumique de type IRM.
- Génération d'arbres selon probabilité (image NG).
- Génération de plusieurs arbres sur le même domaine.
- Base d'apprentissage pour modèle réseau génératif.

Merci de votre attention !

demo IPOL

GitHub OpenCCO team