

OpenCCO: An Implementation of Constrained Constructive Optimization for Generating 2D and 3D Vascular Trees

Bertrand Kerautret¹, Phuc Ngo², Nicolas Passat³, Hugues Talbot⁴, Clara Jaquet⁵
with Adam Germain¹

IPOL/MLBriefs 2023, 12 décembre 2023

Centre Borelli, ENS Paris-Saclay

¹Université Lyon 2, LIRIS

²Université de Lorraine, LORIA

³Université de Reims Champagne Ardenne, CReSTIC

⁴CentraleSupélec, Inria

⁵FindOut Diagnostic AB, Sweden



Plan

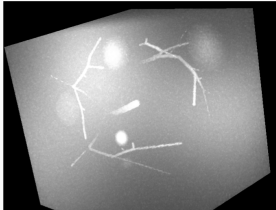
1. Introduction
2. Overview of the main points of the algorithm
3. Implementation
4. Results
5. Conclusion

1. Introduction

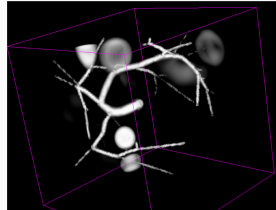
1. Introduction: Context and Initial Motivations (1)

Context related to the ANR project R-VESSEL-X

- Project on the liver vessel analysis.
- PhD of Jonas Lamy on the vessel enhancement filters (2019).
 - enhancement filter on 3D MRI or CT image.
 - few annotated data on MRI liver images.



input images



enhanced filter result

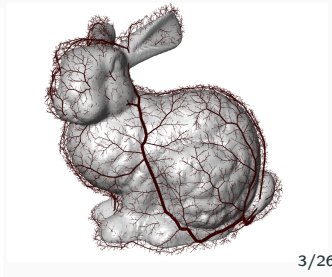
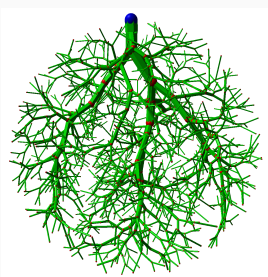
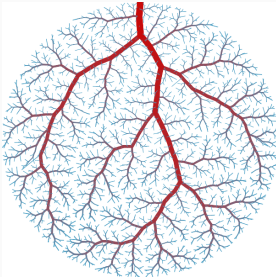
1. Introduction: Context and Initial Motivations (1)

Context related to the ANR project R-VESSEL-X

- Project on the liver vessel analysis.
- PhD of Jonas Lamy on the vessel enhancement filters (2019).
 - enhancement filter on 3D MRI or CT image.
 - few annotated data on MRI liver images.

CCO Algorithm : Constrained Constructive Optimization

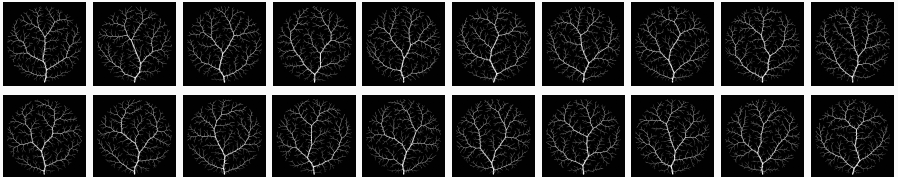
Generate synthetic but realist vascular objects (2D ou 3D)



1. Introduction: Context and Initial Motivations (2)

Numerous interest points

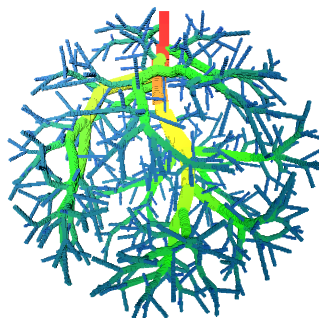
- Synthetic medical images generation.
- Learning based application method.



1. Introduction: Context and Initial Motivations (2)

Numerous interest points

- Synthetic medical images generation.
- Learning based application method.
- Ground truth data sets useful for benchmarks (filtering, geometric analysis).
- Models exploited for simulation or modelling (perfusion/vascular).



1. Introduction: Context and Initial Motivations (2)

Numerous interest points

- Synthetic medical images generation.
- Learning based application method.
- Ground truth data sets useful for benchmarks (filtering, geometric analysis).
- Models exploited for simulation or modelling (perfusion/vascular).

Existing works

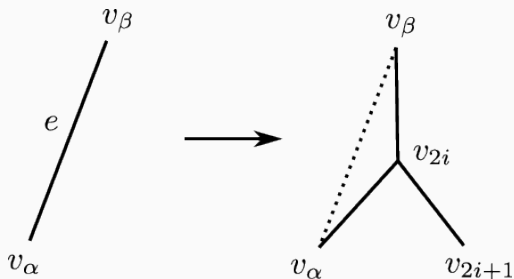
- Schreiner et al. Computer-optimization of vascular trees. IEEE TBE 1993
- VascuSynth : cubic domain, only defined in 3D, simplified optimisation.
- Clara Jaquet PhD Thesis (Hugues Talbot, Paris Est, 2018)
 - Jaquet et al. Generation of Patient-Specific Cardiac Vascular Networks: A Hybrid Image-Based and Synthetic Geometric Model. IEEE TBE 2019
 - Industrial collaboration with HeartFlow company : source code not available.

2. Overview of the main points of the algorithm

Main idea of the algorithm

Incremental construction from n identical steps

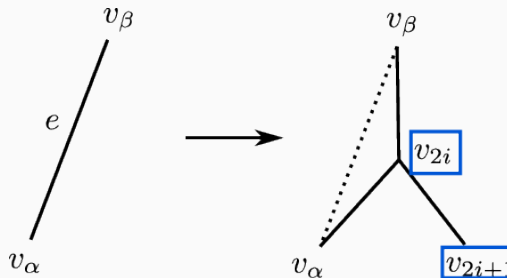
- Adding a terminal point + one incident branch (= one bifurcation)



Main idea of the algorithm

Incremental construction from n identical steps

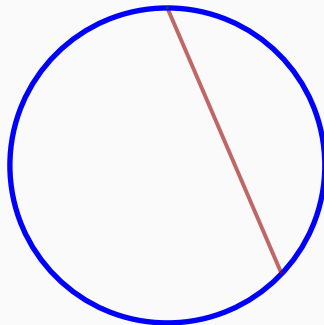
- Adding a terminal point + one incident branch (= one bifurcation)



Main idea of the algorithm

Incremental construction from n identical steps

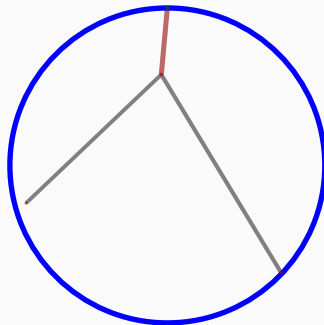
- Adding a terminal point + one incident branch (= one bifurcation)



Main idea of the algorithm

Incremental construction from n identical steps

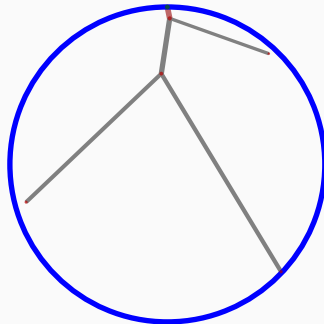
- Adding a terminal point + one incident branch (= one bifurcation)



Main idea of the algorithm

Incremental construction from n identical steps

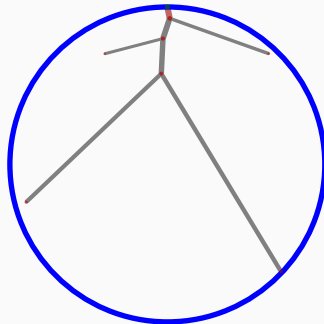
- Adding a terminal point + one incident branch (= one bifurcation)



Main idea of the algorithm

Incremental construction from n identical steps

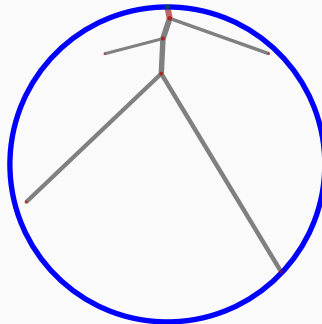
- Adding a terminal point + one incident branch (= one bifurcation)



Main idea of the algorithm

Incremental construction from n identical steps

- Adding a terminal point + one incident branch (= one bifurcation)
- For several candidates $\Rightarrow$ **choose of the best**
 - Selection of one branch to be connected to the new incident branch.
 - Eligibility test (intersection, proximity, domain constraint)
 - **Bifurcation point computation.**
 - Computation (or update) of branch diameters of ascendants branches.



Algorithmic focus

Algorithm (topological aspects)

Input: Number of terminal points $N_{term} \in \mathbb{N}_+^*$

Result: Graph of the vascular tree $(\mathcal{V}, \mathcal{E})$

// graph initialisation

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

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

// Adding segments to the graph

3 **foreach** $i \in \llbracket 1, N_{term} - 1 \rrbracket$ **do**

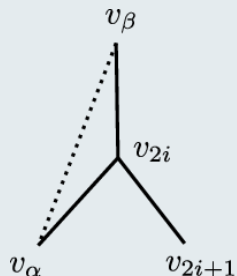
4 Let v_{2i}, v_{2i+1} be two vertices.

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

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



Algorithmic focus

Algorithm (topological aspects)

Input: Number of terminal points $N_{term} \in \mathbb{N}_+^*$

Result: **Graph** of the vascular tree $(\mathcal{V}, \mathcal{E})$

// graph initialisation

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

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

// Adding segments to the graph

3 **foreach** $i \in \llbracket 1, N_{term} - 1 \rrbracket$ **do**

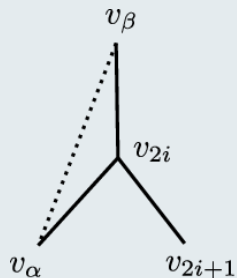
4 Let v_{2i}, v_{2i+1} be two vertices.

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

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

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

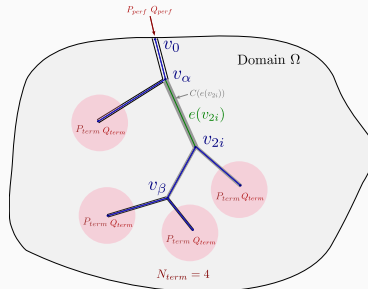
8 Let $(\mathcal{V}, \mathcal{E}) = (\mathcal{V}_{N_{term}-1}, \mathcal{E}_{N_{term}-1})$



Parameters

Algorithm parameters

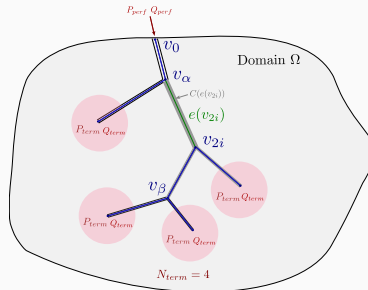
- Dimension $d \in \{2, 3\}$
- Domain $\Omega \subset \mathbb{R}^d$ (convex or non convex) and the root $x_0 \in \Omega$
- Number of terminal points $N_{term} \in \mathbb{N}^*$
- Number of nearest segments $N_{neigh} \in \mathbb{N}^* \Rightarrow$ candidate segment
- Maximal number of nearest segments $N_{max} \in \mathbb{N}^* \Rightarrow$ intersection test



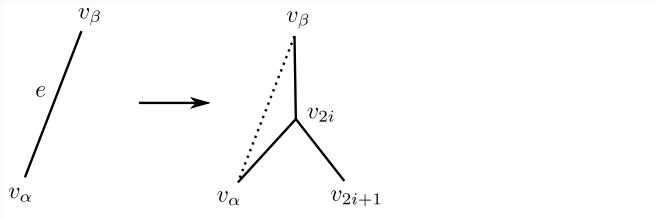
Parameters

Physiological Parameters

- Perfusion volume (A_{perf})
- Input and output Pressure (P_{perf} et P_{term})
- Input and output flow (Q_{perf} et Q_{term})
- Viscosity (μ)
- Murray bifurcation coefficient (γ)



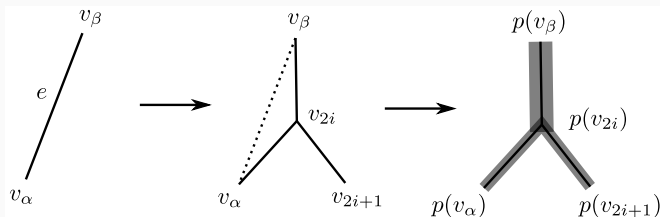
Topological construction but also geometric ...



Geometric features related to the embedding

- Each vertex is embedded in $\mathbb{R}^d$.
- Each edge is associated to a straight segment between two vertices.

Topological construction but also geometric ...



Geometric features related to the embedding

- Each vertex is embedded in $\mathbb{R}^d$.
- Each edge is associated to a straight segment between two vertices.

Constraints related to the embedding :

- Each vertex and segments must be strictly inside Ω
- No segment intersection ($\rightarrow$ width)
- The vertex should be enough far from the tree.
- Each segment must have a minimal length.

From physiological and geometric constraints

To found an optimal solution

- Connexion minimizing the **total volume** of the tree.
- Optimal position of the **bifurcation**.
- Updating the branch diameter
- Eligibility test : intersection, proximity, domain constraints.

Synthesis of the global process (1)

Three distinct phases

- 1. Choice of the random terminal points for the reconstruction :
 - **constraint** : domain, distance, intersection.
 - multiplicity : choice of the attachment segment
→ N_{neigh} nearest segments

Synthesis of the global process (1)

Three distinct phases

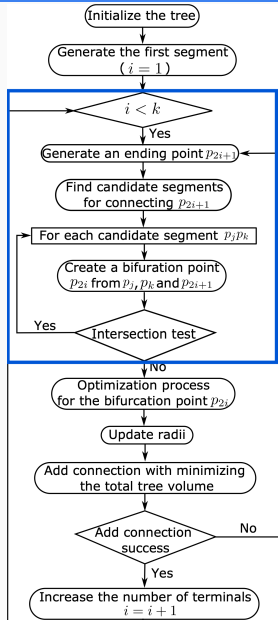
- 1. Choice of the random terminal points for the reconstruction :
 - **constraint** : domain, distance, intersection.
 - multiplicity : choice of the attachment segment
→ N_{neigh} nearest segments
- 2. Optimisation process :
 - **Optimal** position of the bifurcation point.
 - Constraint : domaine, longueur, intersection
→ The N_{max} nearest segments.
 - **Global volume minimization.**

Synthesis of the global process (1)

Three distinct phases

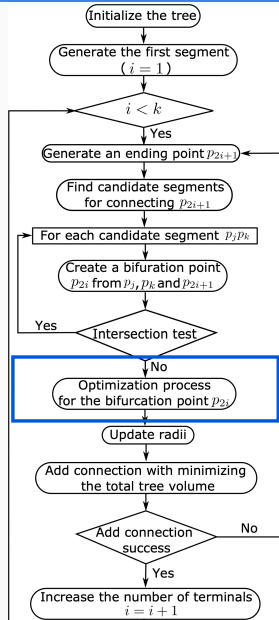
- 1. Choice of the random terminal points for the reconstruction :
 - **constraint** : domain, distance, intersection.
 - multiplicity : choice of the attachment segment
→ N_{neigh} nearest segments
- 2. Optimisation process :
 - **Optimal** position of the bifurcation point.
 - Constraint : domaine, longueur, intersection
→ The N_{max} nearest segments.
 - **Global volume minimization.**
- 3. Updating :
 - Radius of the ascendant branch.
 - Pressure, flow, global volume.

Synthesis of the global process (2)



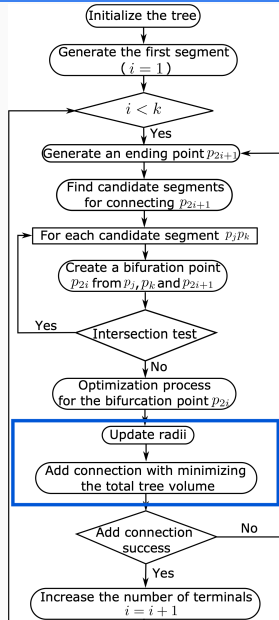
- Step 1 with reconstruction process

Synthesis of the global process (2)



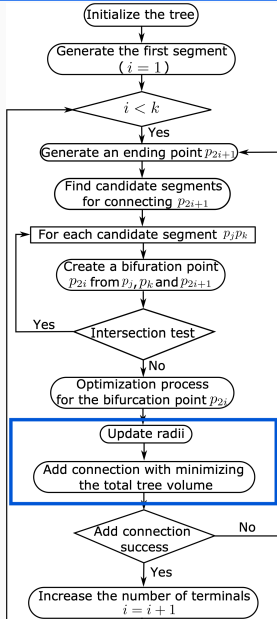
- Step 1 with reconstruction process
- Step 2 with optimisation

Synthesis of the global process (2)



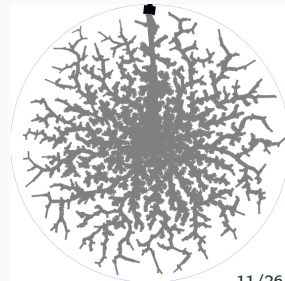
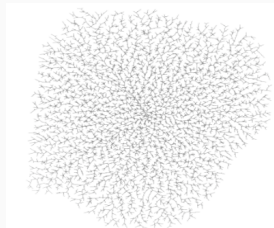
- Step 1 with reconstruction process
- Step 2 with optimisation
- Step 3 update tree

Synthesis of the global process (2)



- Step 1 with reconstruction process
- Step 2 with optimisation
- Step 3 update tree

Importance of optimisation steps:
(first results without optimisation)



3. Implementation

3. Implementation : organisation

Software Organization

- Programming Language : C++
- Compilation tools : `cmake`
- Dependencies :
 - Ceres Solver : non linear solver for the geometric optimization
 - DGtal : 2D/3D domain processing, visualization.
- Diffusion platform: *GitHub*, Continue integration tools: Linux/MacOs
- *Docker* image.

OpenCCO-team / OpenCCO Private

Edit Pins Unwatch 3 Fork 1 Starred 1

<> Code Issues Pull requests Actions Projects Security Insights Settings

ipolSubmit Go to file Add file <> Code About

kerautret Update README.md on Apr 3 476

.github/workflows	github action new runner	4 months ago
.ipol	move graphAnalysis	2 months ago
ExpeIPOL	add expe ipol dir	2 months ago

No description, website, or topics provided.

Readme GPL-3.0 license Activity 1 star

3. Implementation : user interface (1)

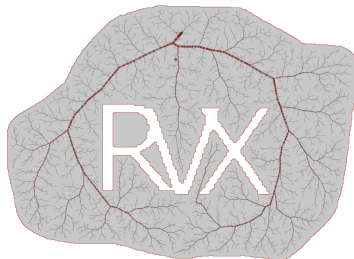
Execution from command lines:

- Main design : defined to script the tree generation.

3. Implementation : user interface (1)

Execution from command lines:

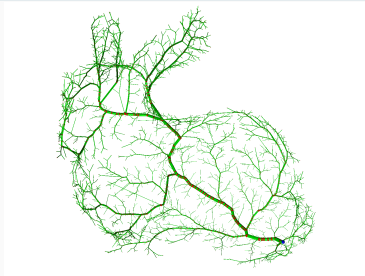
- Main design : defined to script the tree generation.
- `bin/generateTree2D` and `generateTree3D` : generation of the vascular tress 2D/3D
→ export of the tree in xml format.



3. Implementation : user interface (1)

Execution from command lines:

- Main design : defined to script the tree generation.
- bin/generateTree2D and **generateTree3D** : generation of the vascular tress 2D/3D
→ export of the tree in xml format.

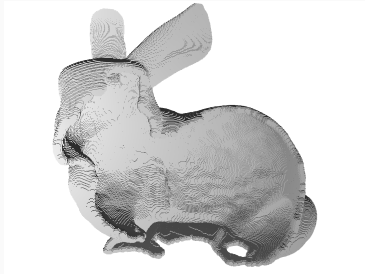


```
generateTree3D -n 3000 -a 20000 -d Samples/bunnyThickBdr.vol -view -m 1 13/26  
-p 143 -107 7
```

3. Implementation : user interface (1)

Execution from command lines:

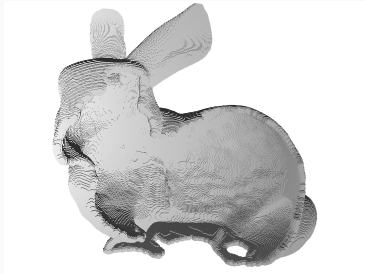
- Main design : defined to script the tree generation.
- bin/generateTree2D and generateTree3D : generation of the vascular tress 2D/3D
→ export of the tree in xml format.



3. Implementation : user interface (1)

Execution from command lines:

- Main design : defined to script the tree generation.
- bin/generateTree2D and generateTree3D : generation of the vascular tress 2D/3D
→ export of the tree in xml format.
- tools/graphAnalysis : tools tp analyse/converse from xml format.
→ also usefull to analyse the xml format generated from VascuSynth.

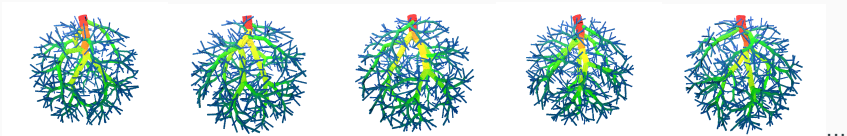


3. Implementation : user interface (2)

Tools can be used together with the DGtalTools-Conbtrib project tools

Example of script defined to generate 100 discretized volumes :

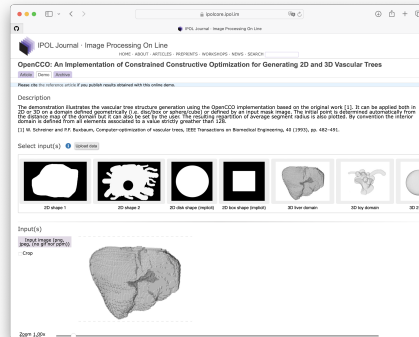
```
#!/bin/bash
typeset -Z3 i
for i in $(seq -w 1 100)
do
    generateTree3D
    xml2graph tree_3D.xml r
    graph2vol -v r_vertex.dat -e r_edges.dat -r r_radius.dat -o vol${i}.nii -g 10
done
```



3. Implémentation : niveau utilisateur (3)

Online demonstration associated to the IPOL paper

- Defined both in 2D and 3D.
- Domains : implicit (geometrical shape), from 2D or 3D images given by user.
- Export available online, xml format.



3. Implementation : developer side (1)

Code structure : two important classes

- **CoronaryArteryTree** : structure of the vascular tree (physiologic parameter, segments and radius rayons, domain définition).

3. Implementation : developer side (1)

Code structure : two important classes

- **CoronaryArteryTree** : structure of the vascular tree (physiologic parameter, segments and radius rayons, domain définition).
- **DomainController** : control the expansion domain of the tree.
Concept that require the implementation of functions `randomPoint()`, `isInside()`, `checkNoIntersectDomain()`

3. Implementation : developer side (1)

Code structure : two important classes

- **CoronaryArteryTree** : structure of the vascular tree (physiologic parameter, segments and radius rayons, domain définition).

- **DomainController** : control the expansion domain of the tree.

Concept that require the implementation of functions `randomPoint()`, `isInside()`, `checkNoIntersectDomain()`

→ Several classes follow the DomainController concept :

- **CircularDomainCtrl** : circular domain.
- **SquareDomainCtrl** : square domain .
- **ImageMaskDomainCtrl** : domain defined from the image associated to a mask.

3. Implementation : developer side (1)

Code structure : two important classes

- **CoronaryArteryTree** : structure of the vascular tree (physiologic parameter, segments and radius rayons, domain définition).
- **DomainController** : control the expansion domain of the tree.
Concept that require the implementation of functions `randomPoint()`, `isInside()`, `checkNoIntersectDomain()`
→ Several classes follow the DomainController concept :
 - **CircularDomainCtrl** : circular domain.
 - **SquareDomainCtrl** : square domain .
 - **ImageMaskDomainCtrl** : domain defined from the image associated to a mask.

Common structure in 2D and 3D

- Class **CoronaryArteryTree** : two template parameters :
 - dimension of the domain 2D/3D.
 - domain controller.
- Domain controller : on template parameter associated to the dimension.

3. Implementation : developer side (2)

Example of minimal code :

Reconstruction in three steps :

1. Types definition and object tree construction:

```
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. Implementation : developer side (2)

Example of minimal code :

Reconstruction in three steps :

1. Types definition and object tree construction:

```
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. Initialization and tree expansion:

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

3. Implementation : developer side (2)

Example of minimal code :

Reconstruction in three steps :

1. Types definition and object tree construction:

```
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. Initialization and tree expansion:

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

3. Result export:

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

3. Implementation : developer side (3)

Extension simple to do:

Ex : define a tree generation guided from the intensity of the source image.

- Implement a new domain controller.
- From parent class ImageMaskDomainCtrl.
- Overload the method RandomPoint().

3. Implementation : developer side (3)

Extension simple to do:

Ex : define a tree generation guided from the intensity of the source image.

- Implement a new domain controller.
- From parent class ImageMaskDomainCtrl.
- Overload the method RandomPoint().

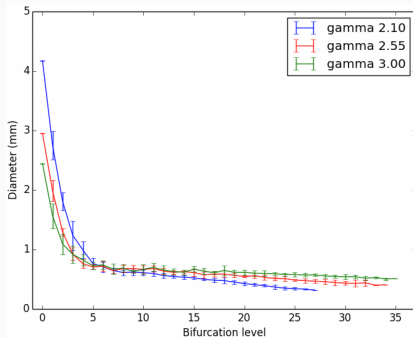
Other perspectives

Make grow two distinct vascular trees on the same domain.

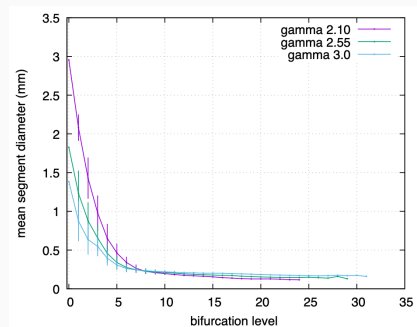
- Domain controller with a reference to another tree.
- Specific overloaded method RandomPoint(), checkNoIntersectDomain().

4. Results

Analyse of the mean diameter



Jaquet



OpenCCO

Execution time

Physiologic parameters

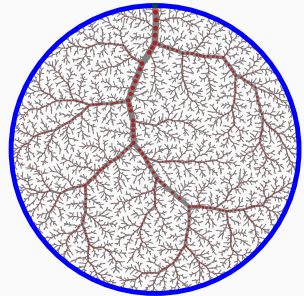
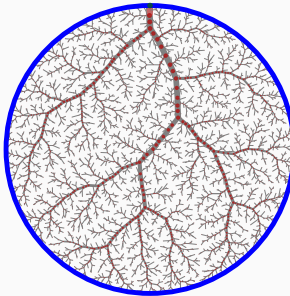
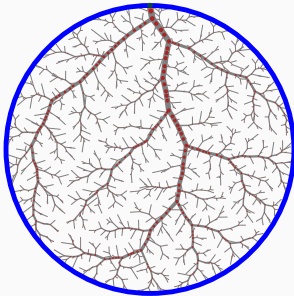
- Perfusion surface / volume : $A_{perf} = 20000 \text{ cm}^2 / \text{cm}^3$
- Perfusion pressure : $P_{perf} = 1.33\text{e}4\text{Pa}$
- Terminal pressure : $P_{term} = 8.33\text{e}3\text{mm}^3/\text{s}$
- Viscosity of blood : $\mu = 3.6\text{cp}$
- Murray bifurcation exponent : $\gamma = 3$

Algorithm parameters

- Number of terminal points : $N_{term} \in \mathbb{N}_+^*$
- Number of segments for the proximity test : $N_{max} = 20$
- Number of segments for the intersection : $N_{neigh} = 20$

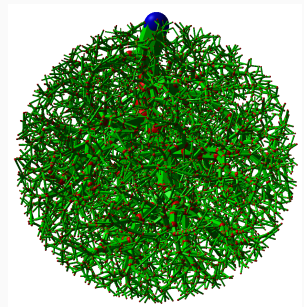
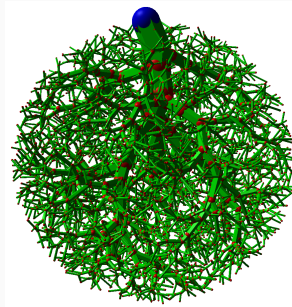
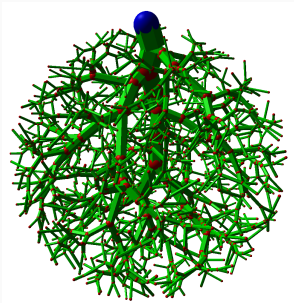
Execution time

Model	N_{term}	Execution time
2D	1000	129 s
	2000	478 s
	3000	1024 s
	4000	1897 s
	5000	3435 s



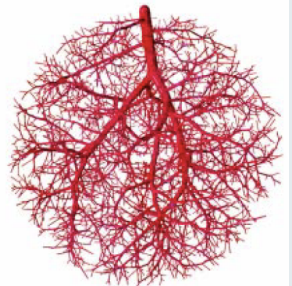
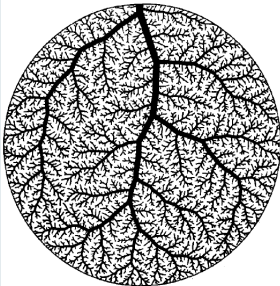
Execution time

Model	N_{term}	Execution time
3D	1000	219 s
	2000	902 s
	3000	1780 s
	4000	3835 s
	5000	5451 s



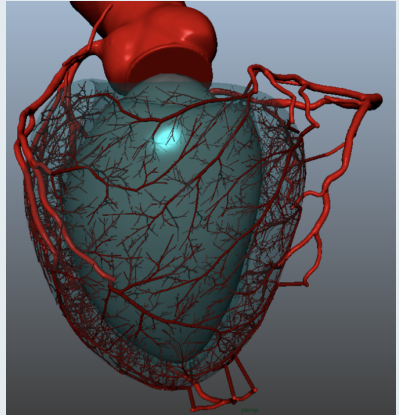
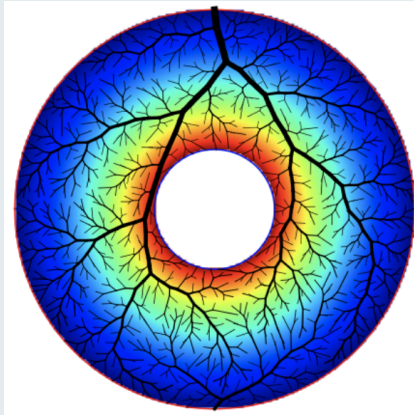
Previous results

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



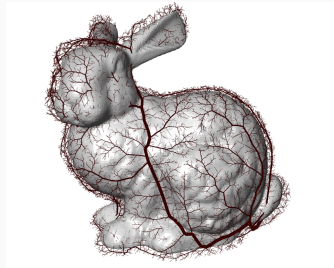
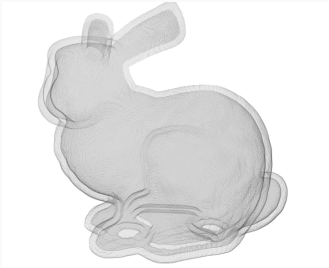
Previous results

CCO 2D & 3D (Jaquet)



Other recent results

CCO on non convex domain



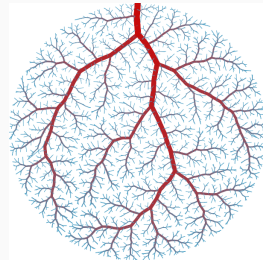
5. Conclusion

Conclusion

- Implementation defined in 2D / 3D in C++.
- Online demonstration and GitHub repository.
- IPOL paper published.
- Comparative study with VascuSynth.

Perspectives (resulting from Master 2 of A. Germain)

- Rendering improvement from resulting flow values.
- Generating rendering images (of type MRI).
- Generating tree from probability map (gray level image).
- Generating several tree from the same domain.
- Learning base for generative for a generative model.



Thanks for your attention !



demo IPOL



GitHub OpenCCO team