Modular priors to shape registration

The deformation module framework

Leander Lacroix

leander.lacroix@ljll.math.upmc.fr

A joint work with Benjamin Charlier, Stanley Durleman, Barbara Gris and Alain Trouvé

Laboratoire Jacques-Louis Lions

November 25, 2019

Table of content

A traditional approach in Shape Analysis

Focus on shape analysis using diffeomorphic transformations.
Large deformations using diffeomorphism.
A diffeomorphism ϕ is built by solving (i.e. integrating) the flow equation:

$$\begin{cases} \phi_0 = \mathrm{Id} \\ \dot{\phi}_t = v_t \circ \phi_t, \qquad t \in [0,1] \end{cases}$$

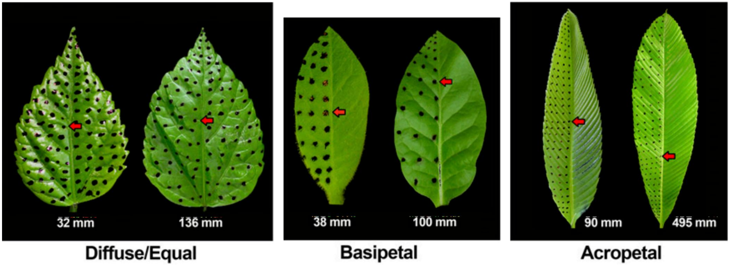
With v_t the vector field at time t.

LDDMM[1]:

- Strong mathematical results
- Mature implementations

[1] Beg, M. F., Miller, M. I., Trouvé, A., Younes, L. (2005). Computing large deformation metric mappings via geodesic flows of diffeomorphisms. International journal of computer vision, 61(2), 139-157.

Example: different growth patterns of leaves



[Gupta, M. D., Nath, U. (2015). Divergence in patterns of leaf growth polarity is associated with the expression divergence of miR396. The Plant Cell, tpc-15.]

Non parametric growth in the case of the basipetal pattern

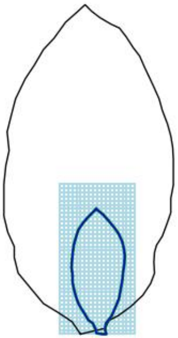


Figure: $t = 0$

Non parametric growth in the case of the basipetal pattern

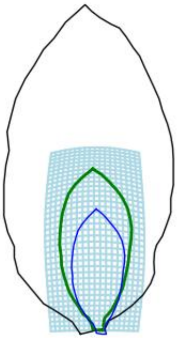


Figure: $t = 0.25$

Non parametric growth in the case of the basipetal pattern

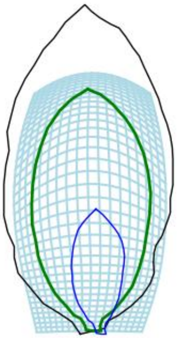


Figure: $t = 0.5$

Non parametric growth in the case of the basipetal pattern

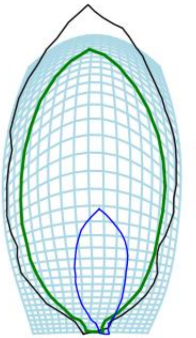


Figure: $t = 0.75$

Non parametric growth in the case of the basipetal pattern

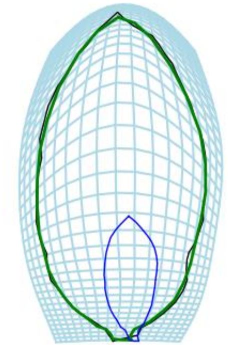
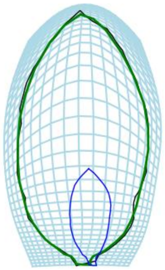


Figure: $t = 1$

Non parametric growth in the case of the basipetal pattern: comparison



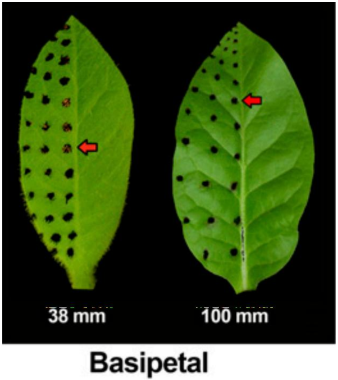


Figure: Caption

Prior work in parametric deformations

Sparse LDDMM (Deformetrica) [S. Durrleman, M. Prastawa, G. Gerig, and S. Joshi. Optimal data-driven sparse parameterization of diffeomorphisms for population analysis. In Information Processing in Medical Imaging , pages 123-134. Springer, 2011]

Higher order momentum [S. Sommer M. Nielsen, F. Lauze, and X. Pennec. Higher-order momentum distributions and locally affine lddmm registration. SIAM Journal on Imaging Sciences, 2013]

GRID [U. Grenander , A. Srivastava , S. Saini. A pattern-theoric characerization of biological growth. IEEE, 2007]

Poly-affine [V. Arsigny, X. Pennec, N. Ayache, 2005. Polyrigid and Polyaffine Transformations: A Novel Geometrical Tool to Deal with Non-rigid Deformations – Application to the Registration of Histological Slices. Medical Image Analysis 9, 507–523]

Diffeons [L. Younes. Constrained diffeomorphic shape evolution. Foundations of Computational Mathematics, 2012.]

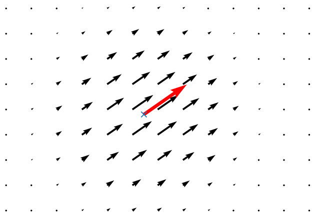
Table of content

Deformation modules: definition

Generates a vector field of specific type chosen by the user

Composed of:

- Geometrical descriptors
- Controls
- Field generator
- Cost function

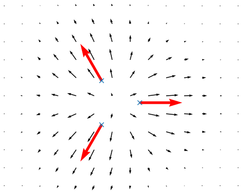


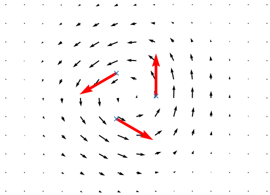
Local translations

Vector field generated by a sum of local translations supported by a gaussian kernel.

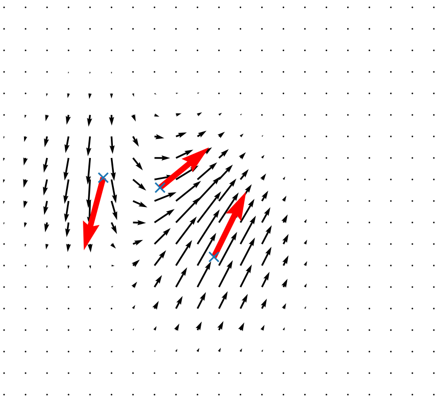
Constrained Translation Generator (CTG)
CTG modules can be used to define:

- Sum of local translation whose direction is constrained
- Local scaling module
- Local rotation module

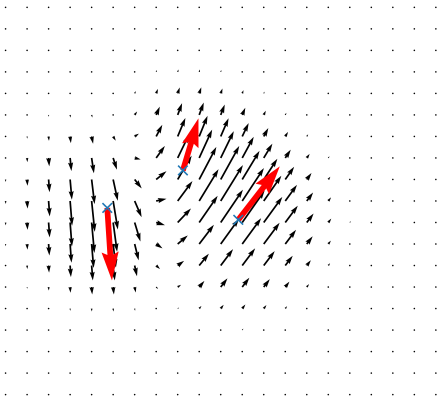




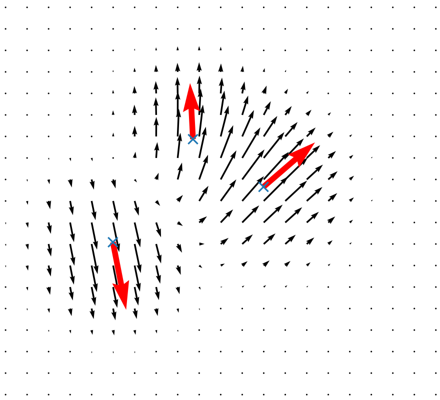
Local translations, a visual example



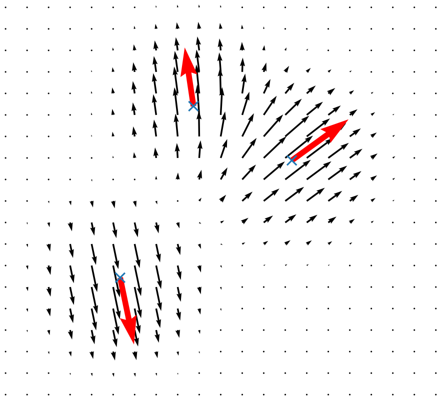
Local translations, a visual example



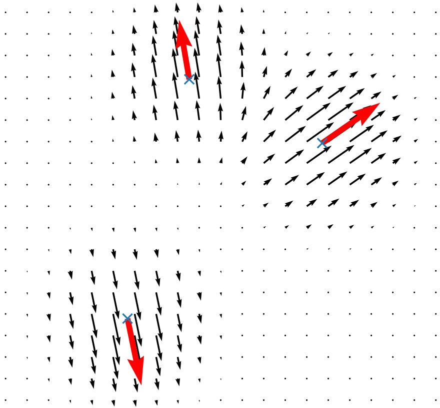
Local translations, a visual example



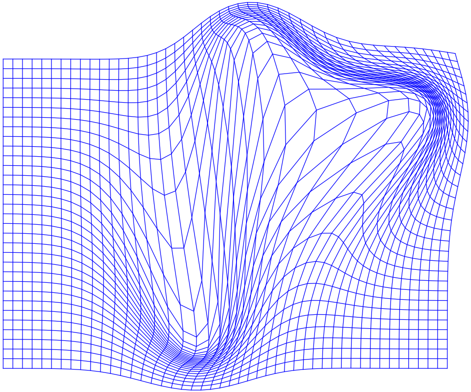
Local translations, a visual example



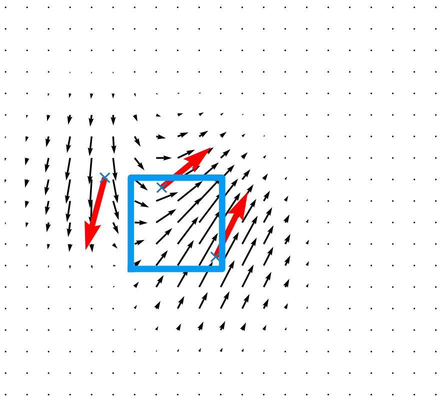
Local translations, a visual example



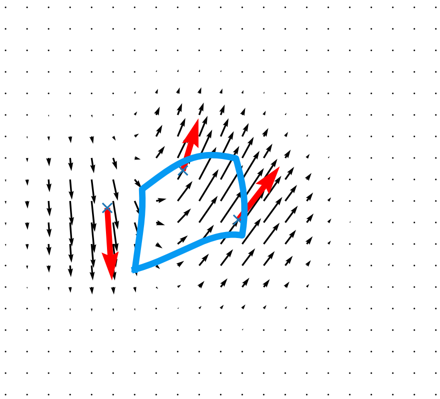
Local translations, a visual example



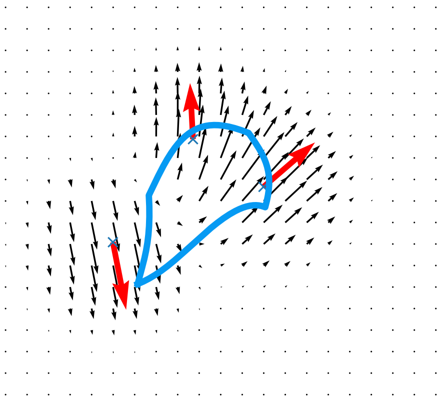
Silent module, a visual example



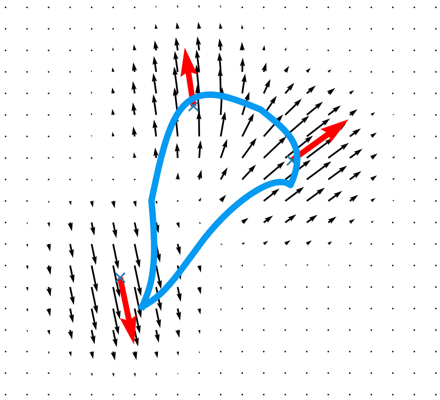
Silent module, a visual example



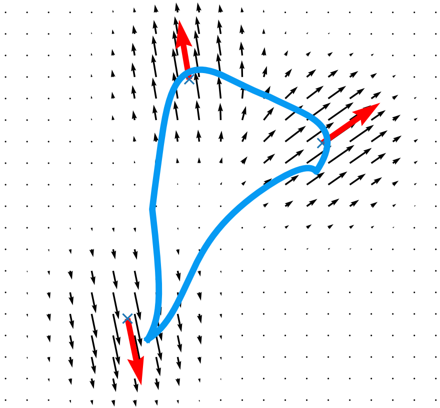
Silent module, a visual example



Silent module, a visual example

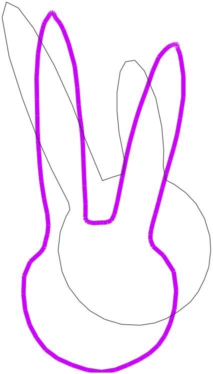


Silent module, a visual example



The registration problem

Deformation that transforms a source object q_S into a target object q_T in shape space



Energy minimisation

Minimisation of an energy functional $\mathcal{J}$:

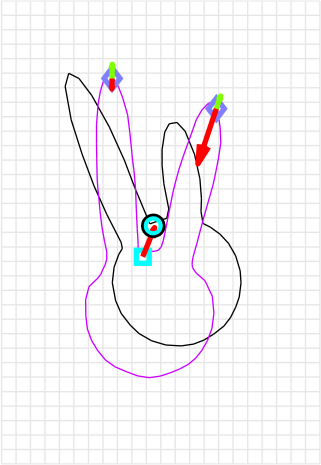
$$\mathcal{J}(\phi; q_S, q_T) = \mathcal{U}(\phi; q_S, q_T) + \mathcal{R}(\phi)$$

With,

- $\mathcal{U}$: similarity between the deformed source and the target
- $\mathcal{R}$: regularity of the deformation ϕ, i.e. its cost

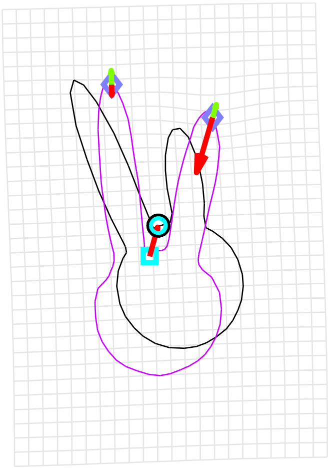
Matching bunnies

t=0

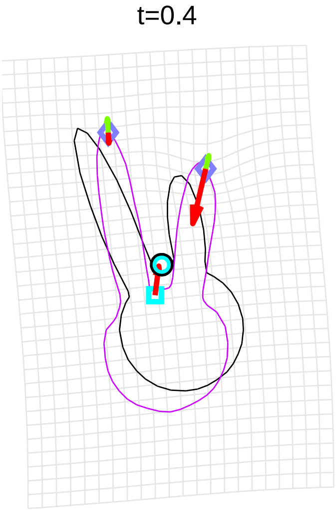


Matching bunnies

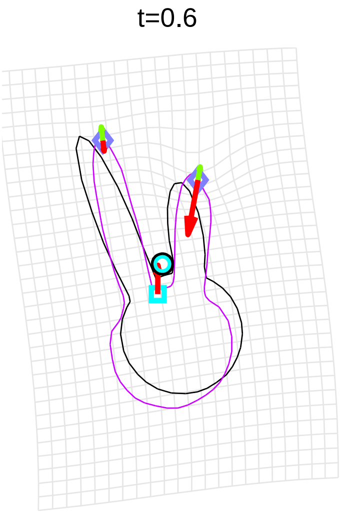
t=0.2



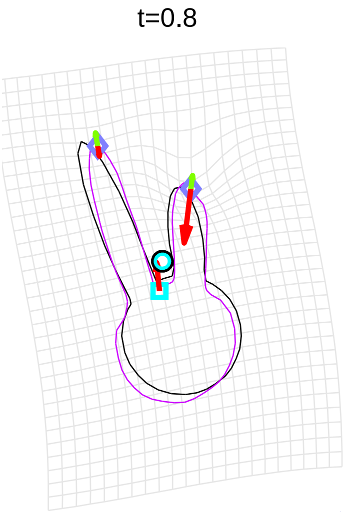
Matching bunnies



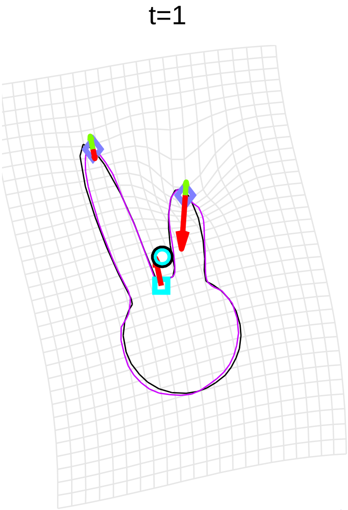
Matching bunnies



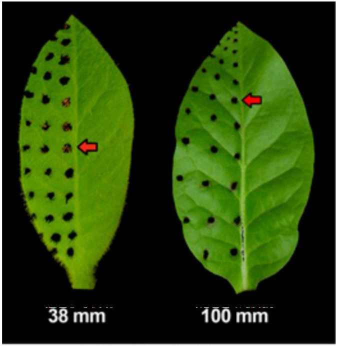
Matching bunnies



Matching bunnies



Back to the leaf



Basipetal

Implicit modules

Implicit definition:

$$\zeta_q(h) = \underset{v}{\operatorname{argmin}}\{\mathrm{Cons}(v, h) + \eta|v|_V^2\}$$

Example: implicit modules of order 0:

$$\mathrm{Cons}(v, x, h) = |v \cdot x - h|_V^2$$

- Similar to translation module
- Better numerical stability

Implicit modules of order 1

Goal: model growth.
Let's define:

$$\mathrm{Cons}(v, x, h) = \sum_i |\mathcal{E}(v, x_i) - S_i h|^2 ,$$

with:

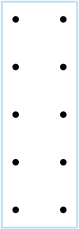
$$\mathcal{E}(v, x_i) = \frac{Dv(x_i) + Dv(x_i)^T}{2},$$

the infinitesimal deformation tensor.

S_i defines the growth.

Isotropic growth

Isotropic growth



Isotropic growth

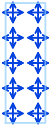


Figure: $t = 0$

$$S_i = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$$

Isotropic growth

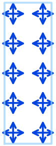


Figure: $t = 0.25$

$$S_i = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$$

Isotropic growth

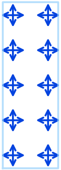


Figure: $t = 0.5$

$$S_i = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$$

Isotropic growth

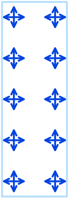


Figure: $t = 0.75$

$$S_i = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$$

Isotropic growth

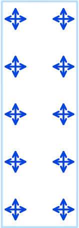


Figure: $t = 1$

$$S_i = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$$

Constant volume growth

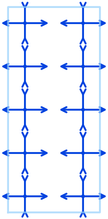


Figure: $t = 0$

$$S_i = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$$

Constant volume growth

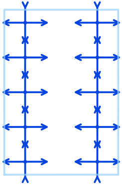


Figure: $t = 0.25$

$$S_i = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$$

Constant volume growth

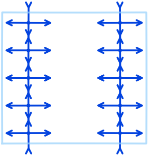


Figure: $t = 0.5$

$$S_i = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$$

Constant volume growth

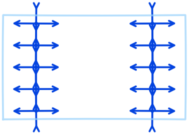


Figure: $t = 0.75$

$$S_i = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$$

Constant volume growth

$$S_i = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$$

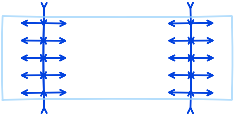
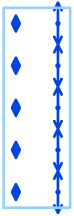


Figure: $t = 1$

Bending



$$S_i = \begin{pmatrix} 0 & 0 \\ 0 & a\, x_i \end{pmatrix}$$

Bending

$$S_i = \begin{pmatrix} 0 & 0 \\ 0 & a\, x_i \end{pmatrix}$$

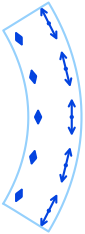


Figure: $t = 0.25$

Bending !

$$S_i = R_i\, C_i\, R_i^T$$

$$R_i(t) = \phi_t \cdot R_i(t = 0)$$

Bending

$$S_i = \begin{pmatrix} 0 & 0 \\ 0 & a\,x_i \end{pmatrix}$$

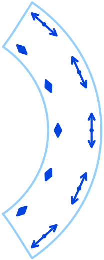


Figure: $t = 0.5$

Bending !

$$S_i = R_i\, C_i\, R_i^T$$

$$R_i(t) = \phi_t \cdot R_i(t=0)$$

Bending

$$S_i = \begin{pmatrix} 0 & 0 \\ 0 & a\,x_i \end{pmatrix}$$

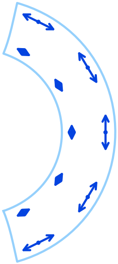


Figure: $t = 0.75$

Bending !

$$S_i = R_i\, C_i\, R_i^T$$

$$R_i(t) = \phi_t \cdot R_i(t=0)$$

Bending

$$S_i = \begin{pmatrix} 0 & 0 \\ 0 & a\,x_i \end{pmatrix}$$

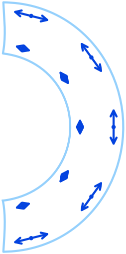


Figure: $t = 1$

Bending !

$$S_i = R_i\, C_i\, R_i^T$$

$$R_i(t) = \phi_t \cdot R_i(t=0)$$

Back to the first example: using an implicit module

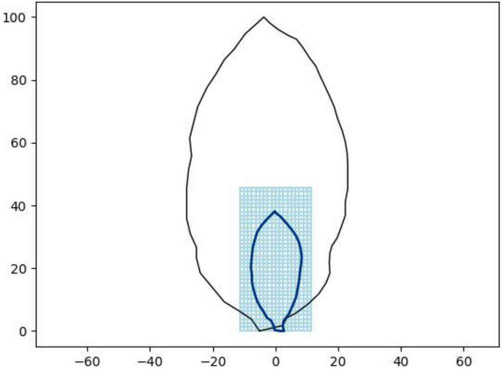


Figure: $t = 0$

Back to the first example: using an implicit module

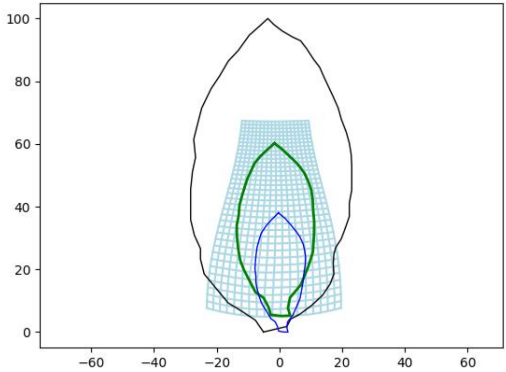


Figure: $t = 0.25$

Back to the first example: using an implicit module

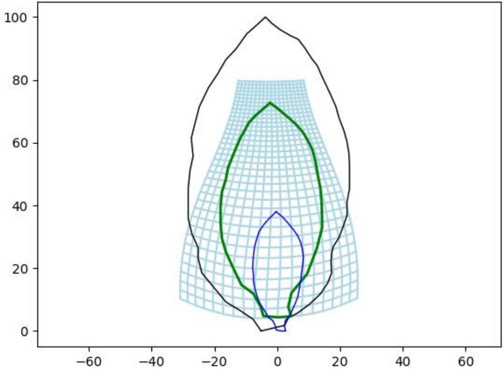


Figure: $t = 0.5$

Back to the first example: using an implicit module

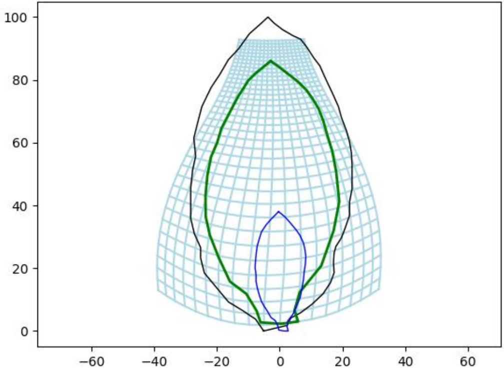


Figure: $t = 0.75$

Back to the first example: using an implicit module

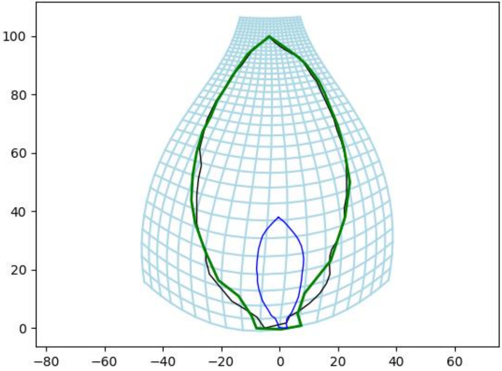
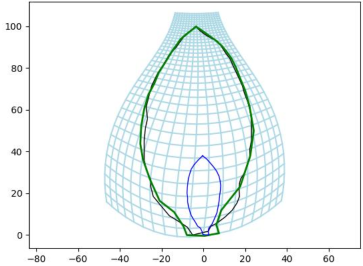
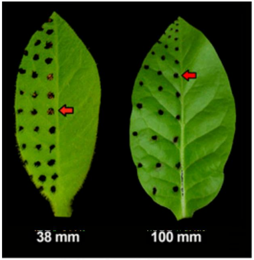


Figure: $t = 1$

Back to the first example: using an implicit module





Basipetal

Table of content

Our implementation of the deformation module framework

Written in Python using Pytorch.

- Automatic differentiation
- GPU computation

KeOps[2] support.

GemLoss[3] support.

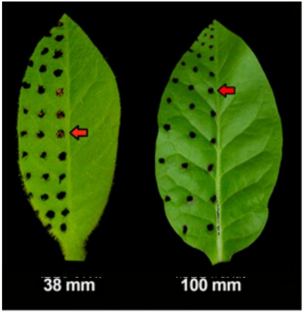
Works in 2D and 3D

Available on our GitLab:

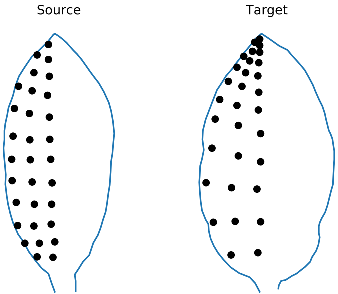
- `plmlab.math.cnrs.fr/gris/implicitmodules`

[2] `github.com/getkeops/keops`
[3] `github.com/jeanfeydy/geomloss`

Basipetal growth: source and target



Basipetal



Basipetal growth: defining modules

```
nu0, nu1 = 0.001, 0.001
sigma0, sigma1 = 10., 100.

global_trans = GlobalTranslation.build(2)

implicit0 = Implicit0.build(2, pos0.shape[0], sigma0,
    nu0, gd=pos0, backend='torch')

implicit1 = Implicit1.build(2, pos1.shape[1], sigma1,
    nu1, C, gd=(pos1, pos1_r), backend='torch')
```

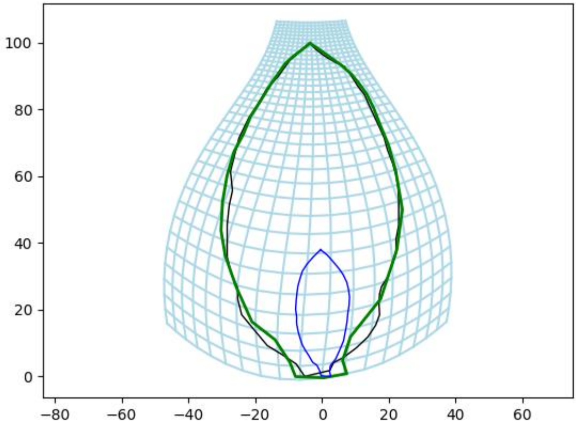
Basipetal growth: using KeOps on GPU

```
nu0, nu1 = 0.001, 0.001
sigma0, sigma1 = 10., 100.

global_trans = GlobalTranslation.build(2)

implicit0 = Implicit0.build(2, pos0.shape[0], sigma0,
    nu0, gd=pos0, backend='keops')

implicit1 = Implicit1.build(2, pos1.shape[1], sigma1,
    nu1, C, gd=(pos1, pos1_r), backend='keops')

global_trans.to('cuda')
implicit0.to('cuda')
implicit1.to('cuda')
```

Basipetal growth: matching

```
model = ModelPointsRegistration(
            [curve_source],
            [global_trans, implicit0, implicit1],
            [VarifoldAttachment(2, [10., 50.])])

fitter = ModelFittingScipy(model, 1.)
costs = fitter.fit([curve_target], 100)
```

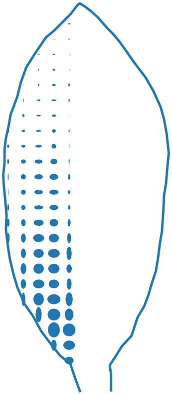
Basipetal growth: matching with a model of growth



Learning the growth pattern

```
model = ModelPointsRegistration(
            [curve_source, dots_source],
            [global_trans, implicit0, implicit1],
            [VarifoldAttachment(2, [10., 50.])],
            other_parameter=[implicit1.C])

fitter = ModelFittingScipy(model, 1.)
costs = fitter.fit([pos_target, dots_target], 100)
```

Learning the growth pattern: result



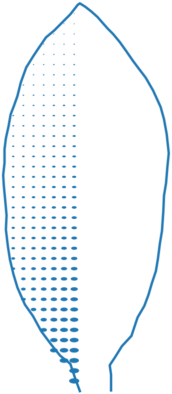
Learning a model of growth pattern

```
coeffs = zeros(3, 2)
coeffs[0] = ones(2)

model = ModelPointsRegistration(
            [curve_source, dots_source],
            [global_trans, implicit0, implicit1],
            [VarifoldAttachment(2, [10., 50.])],
            other_parameter=[coeffs],
            precompute_callback=funComputeC)

fitter = ModelFittingScipy(model, 1.)
costs = fitter.fit([curve_target, dots_source], 100)
```

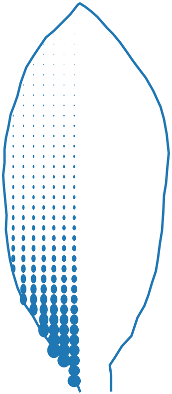
Learning a model of growth pattern

```
def pol_linear(pos, coeffs):
    return coeffs[0] +
        coeffs[1]*pos[:, 0] + coeffs[2]*pos[:, 1]

def funComputeC(init_states, modules, parameters):
    coeffs = parameters['C']
    pos = modules['implicit1'].position
    modules['implicit1'].C = pol_linear(pos, coeffs)
```

Learning a linear model of growth pattern

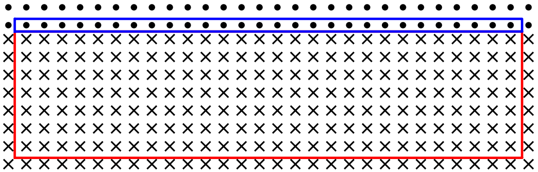


Learning a quadratic model of growth pattern



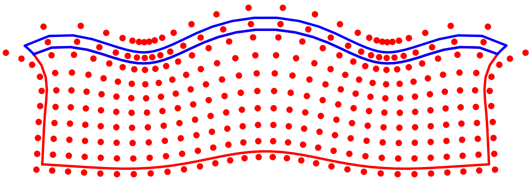
Layered model

Layered model



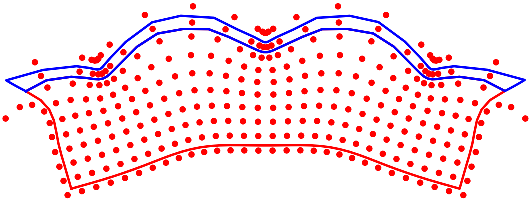
Layered model, defining growth constants, 2 periods

Layered model, shooting for 2 periods

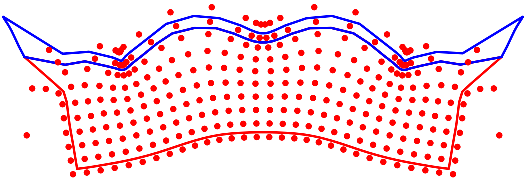


Layered model, defining growth constants, 3 periods

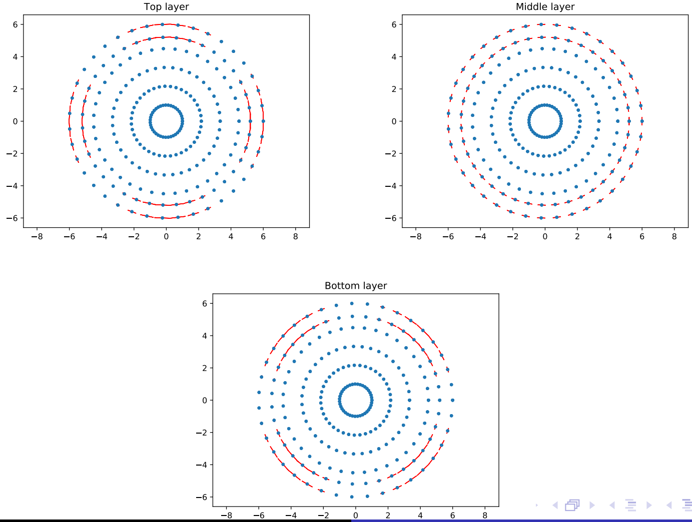
Layered model, shooting for 3 periods



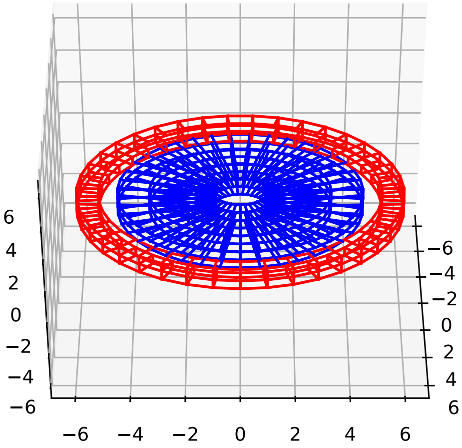
Layered model, shooting for combined 2 and 3 periods



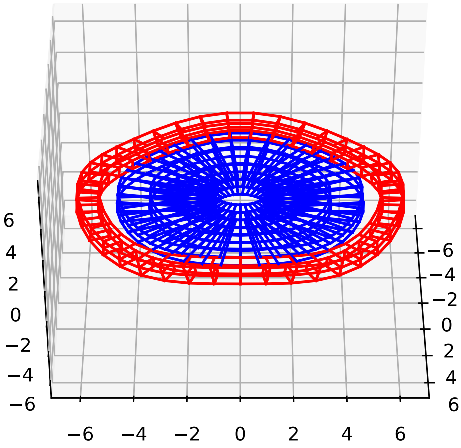
3D layered plate model: growth constant



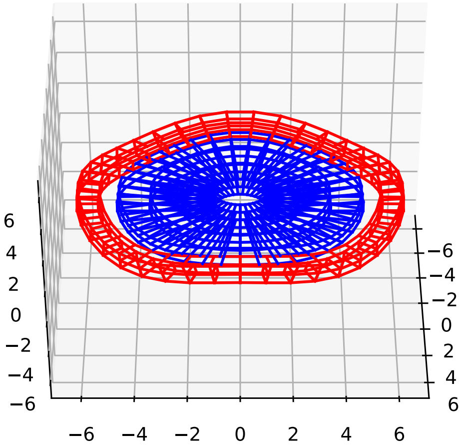
3D layered plate model



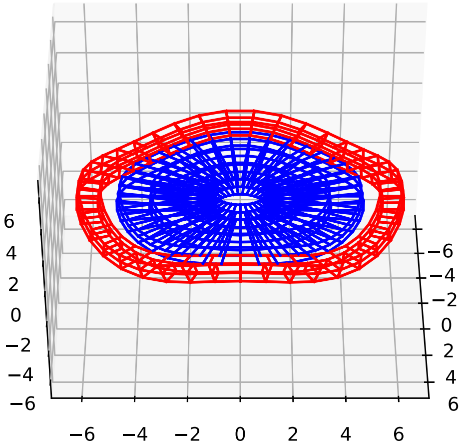
3D layered plate model



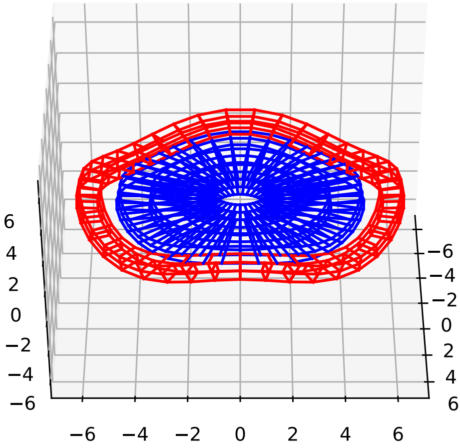
3D layered plate model



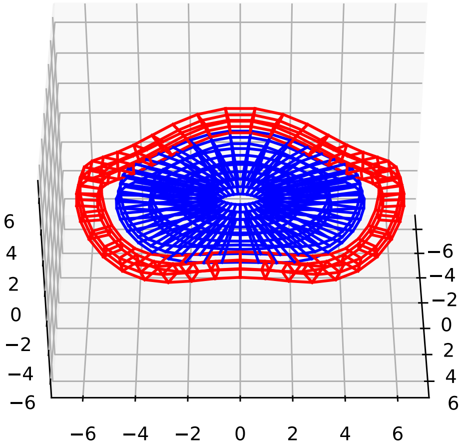
3D layered plate model



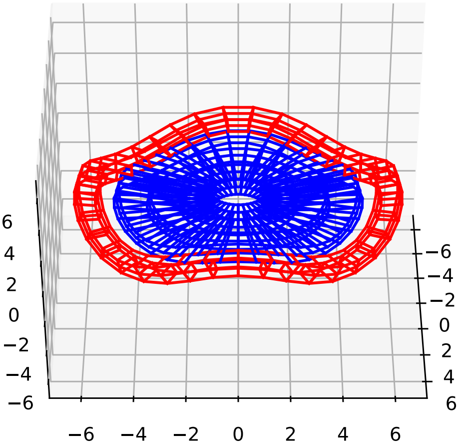
3D layered plate model



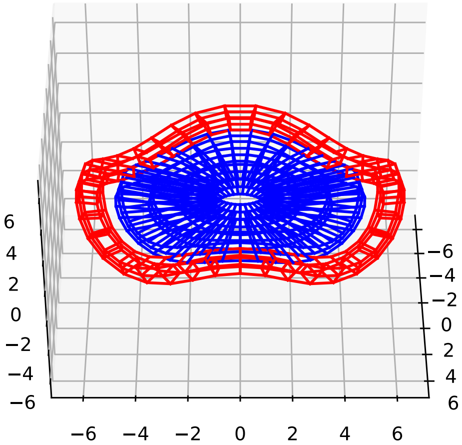
3D layered plate model



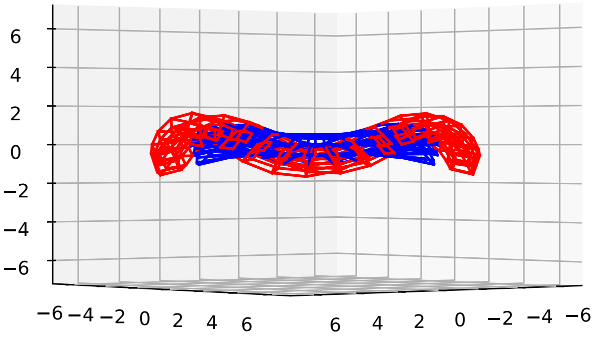
3D layered plate model



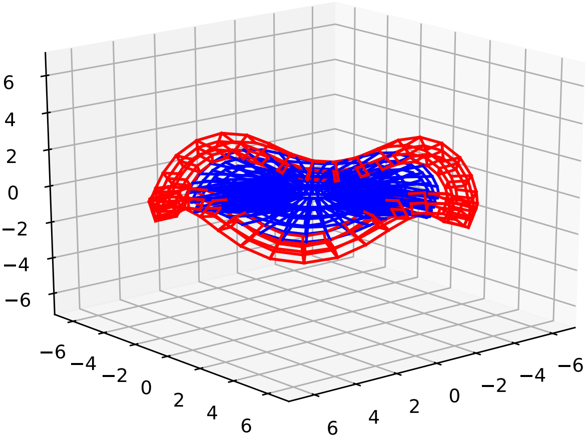
3D layered plate model



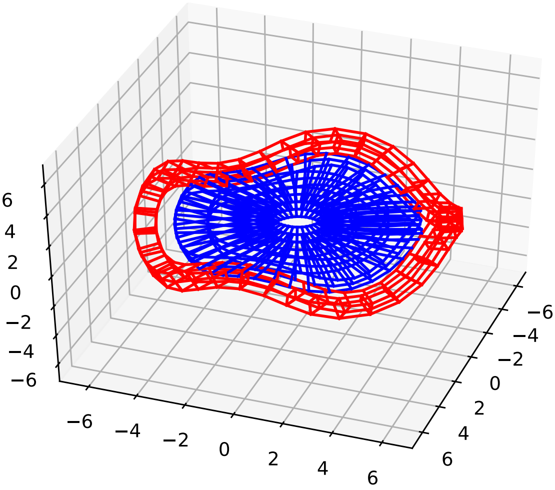
3D layered plate model



3D layered plate model



3D layered plate model



Roadmap

- Finalise work on Atlas
- Better memory usage
- The user should be able to choose other kernels (or even define custom ones)
- Fix some performance issues
- Documentation, user friendly examples
- Better 3D tools (utility functions, plotting, ...)

Conclusion

- New tools to incorporate priors into deformations
- Implicit deformations
    - Elastic behaviour
- Working implementation

`plmlab.math.cnrs.fr/gris/implicitmodules`