Vector field reconstruction from sparse samples with applications

Marcos Lage, Fabiano Petronetto, Afonso Paiva, Hélio Lopes, Thomas Lewiner and Geovan Tavares

Department of Mathematics — Pontifícia Universidade Católica — Rio de Janeiro — Brazil
{mlage, fbipetro, apneto, lopes, tomlew, tavares}@mat.puc--rio.br.

Abstract. We present a novel algorithm for $2D$ vector field reconstruction from sparse set of points–vectors pairs. Our approach subdivides the domain adaptively in order to make local piecewise polynomial approximations for the field. It uses partition of unity to blend those local approximations together, generating a global approximation for the field. The flexibility of this scheme allows handling data from very different sources. In particular, this work presents important applications of the proposed method to velocity and acceleration fields' analysis, in particular for fluid dynamics visualization.

Keywords: ***Vector field reconstruction. Partition of unity. Function approximation.***

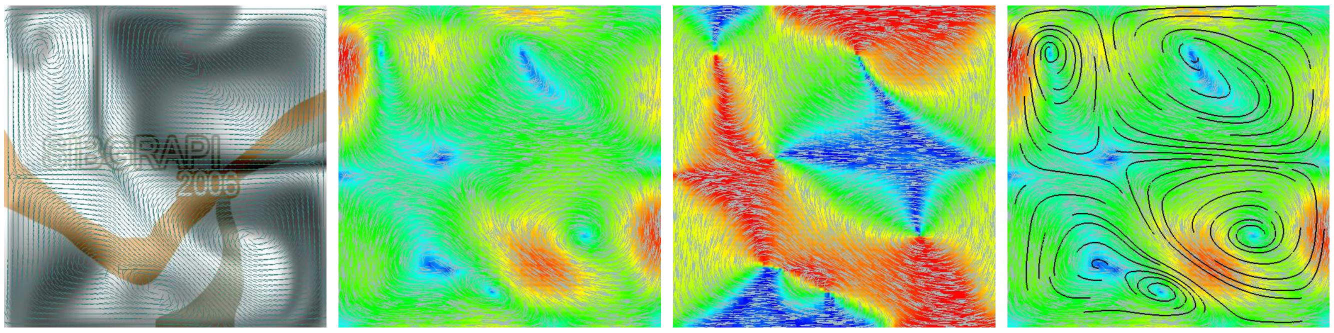


Figure 1: *Velocity field approximation of a smoke simulation: samples of the original field, magnitude, phase and integral curves of the approximated field.*

1 Introduction

The fundamental tools of classical physics are built on vector fields: the motion of an object is represented by its velocity vector field, and the fundamental law of mechanics equals the acceleration vector field to the external force vector field. These fields appear generally in computer graphics as measures of real phenomena, for example using Particle Image Velocimetry methods, or as results of physical simulations, such as fluid simulations through Smoothed Particle Hydrodynamics.

Particle Image Velocimetry (PIV) became an important and active research field in mechanical engineering. It is concerned with the quantitative investigation of fluids by imaging techniques [11]. PIV systems captures the light scattered by small particles in a flow, and extracts from the image sequence a set of points equipped with their estimated velocity vectors. The reconstruction of the velocity field from these maps has several applications, in particular to modern aerodynamics and hydrodynamics research [13].

Smoothed particle hydrodynamics (SPH) has been recognized as a flexible mesh free method for computational fluid dynamics simulations [17]. In SPH the fluid is modeled as a collection of particles, which move under the influence of hydrodynamic and external forces. Each portion of fluid is represented by a particle with attributes, among which the velocity and the acceleration vectors. In the field of computer graphics, SPH has been applied for deformable models [4], free surface flows [8] and blood simulation [9], among others.

Motivation. On one hand, a PIV velocity map contains a set of points with their velocity vectors. Each point corresponds to a pixel on the image. The resolution of this map is thus defined by the resolution of the camera used in the acquisition process. On the other hand, the set of particles at a given time t on a SPH simulation is completely unstructured. For both problems, we aim at inferring a differentiable vector field defined on the whole region of experimentation. This field would not only improve visualization, but also help

Preprint MAT. 12/06, communicated on May 14^{th}, 2006 to the Department of Mathematics, Pontifícia Universidade Católica — Rio de Janeiro, Brazil. The corresponding work was published in the proceedings of the Sibgrapi 2006, pp. 297–304. IEEE Press, 2006.

in analyzing the field structure, for example, by identifying the existence of vortices.

Contributions. This work proposes a novel algorithm for 2D vector field reconstruction considering as input unstructured sets of points–vector pairs (formalized at section 3 *Vector field and polynomial approximation*). Our approach uses least squares techniques (detailed at section 4 *Local vector field approximation*) on a multiresolution grid to generate local approximations. After that, we combine these approximations through partitions of unity, obtaining a global, smooth description of the vector field (as detailed at section 5 *From local to global vector field evaluation*). This approach extends previous approximation techniques to vector field. Comparing to purely visual techniques such as texture interpolations, we are not restricted to regular grid or to low order approximations. Moreover, we improve the numerical stability of the approximation using ridge regression techniques. We conclude this work with important applications to visualization and analysis of the fluid velocity field.

2 Previous and related works

In this work we combine three different techniques: *least squares fitting*, *ridge regression* and *partition of unity*.

Least squares fitting is a mathematical procedure for finding the best approximation function f to a given set of points. To do so, it minimizes the sum of the squared residuals of the points to function f [7]. It has several applications in the fields of computer graphics [18], geometric modeling [14], image processing [16] and computer vision [6]. Several works use least squares to reconstruct planar curves [2] and surfaces [19] from sparse points. Here, we use this mathematical framework to build local approximations for the vector field, minimizing its residual on the given point–vector pairs.

Ridge regression is a technique that is frequently used by statisticians to remove the collinearity of the input points [5]. This technique avoids computationally expensive iterations of pseudo–inverse approaches and improves the least–square solution even if the input points are not collinear. Tasdizen et al. [3] applied such technique to improve the least squares algebraic curve fitting from sparse points in the plane. Along the same lines, we will use *ridge regression* to regularize ill-conditioned linear systems produced by our least squares problem.

Partition of unity [1, 15] is a very useful mathematical tool to combine local approximations in order to construct a global one. Important properties such as the global maximal error and the convergence order could be inherited from the local approximations. Ohtake et al. in [10] proposed a partition of unit based multiresolution method, called *Multi-level Partition of Unity* (MPU), that reconstructs an implicit surface approximation from a set of sparse sample points and normals in $\mathbb{R}^3$. This work extends their ideas in order to build a multiresolution scheme for vector field reconstruction. Although our work is for planar fields, it can be easily extended to 3D.

3 Vector field and polynomial approximation

Sampled vector field. We will consider a set of points $\mathcal{P} = \{\mathbf{p}_1, \mathbf{p}_2, ..., \mathbf{p}_n\}$, where each point $\mathbf{p}_i = (x_i, y_i) \in \Omega \subset \mathbb{R}^2$ base a vector $\mathbf{v}_i$, and denote the set of vectors $\{\mathbf{v}_1, \mathbf{v}_2, ..., \mathbf{v}_n\}$ by $\mathcal{V}$. We will suppose that each vector $\mathbf{v}_i$ is sampled from a differentiable vector field $\mathcal{F} : \Omega \subset \mathbb{R}^2 \to \mathbb{R}^2$ at $\mathbf{p}_i$: $\mathbf{v}_i = \mathcal{F}(\mathbf{p}_i)$. A *vector field* is a map $\mathcal{F} : \mathbb{R}^2 \to \mathbb{R}^2$ that assigns a vector $\mathcal{F}(\mathbf{p}) = (\ P(\mathbf{p}), Q(\mathbf{p})\ )$ to each point $\mathbf{p} \in \mathbb{R}^2$. The functions $P : \mathbb{R}^2 \to \mathbb{R}$ and $Q : \mathbb{R}^2 \to \mathbb{R}$ are called the *coordinate functions* of the vector field $\mathcal{F}$. We aim at inferring an approximation of $\mathcal{F}$ on region Ω.

Polynomial function. We will approximate each coordinate function of $\mathcal{F}$ by a *bivariate polynomial* of a fixed degree d, i.e. $\mathbf{F}(x,y) = (\ P_d(x,y), Q_d(x,y)\ )$ with:

$$P_d(x,y) = \sum_{0 \le j+k}^{d} a_{j,k} x^j y^k, \quad Q_d(x,y) = \sum_{0 \le j+k}^{d} b_{j,k} x^j y^k.$$

Notation. Since polynomial functions are the main mathematical object used in this paper, it is convenient to define a suitable notation. We will use the matrix notation of Tasdizen et al. [2]:

$$P_d(x,y) = \mathbf{w}_{(x,y)}^t \mathbf{a}, \quad Q_d(x,y) = \mathbf{w}_{(x,y)}^t \mathbf{b} \qquad (1)$$

where column $\mathbf{a} \in \mathbb{R}^l$ contains the coefficients $[a_{j,k}]$ of P_d for $j + k \le d$:

$$\mathbf{a} = \left[a_{0,0}\ a_{1,0} \ldots a_{d,0}\ a_{0,1} \ldots a_{d-1,1}\ a_{0,2} \ldots a_{d-2,2} \ldots\ a_{0,d}\right]^t$$

and column $\mathbf{w}_{(x,y)} \in \mathbb{R}^l$ contains the monomials of degree less than d:

$$\mathbf{w}_{(x,y)} = \left[1\ x \ldots x^d\ y \ldots (x^{d-1}y)\ y^2 \ldots (x^{d-2}y^2) \ldots y^d\right]^t$$

The dimension l of columns $\mathbf{a}$ and $\mathbf{w}_{(x,y)}$ is the number of coefficients of $P_d : \#P_d = \frac{(d+1)(d+2)}{2}$.

4 Local vector field approximation

We aim at inferring a polynomial vector field $\mathbf{F}(x,y)$= $(\ P_d,\ Q_d\ )$ that best approximates each sample vector $\mathbf{v}_i$ at $\mathbf{p}_i$. This section introduces the least square technique we use for minimizing locally the approximation error between $\mathbf{F}(\mathbf{p}_i)$ and $\mathbf{v}_i$ (section 4(a) *Classical least squares fitting*). Our approximation also incorporates eventual knowledge of the vector field derivative at the sample points (section 4(b) *Acceleration fitting*). We improve the numerical stability of this local minimization using ridge regression techniques (section 4(c) *Ridge regression*). These techniques are finally combined and weighted using two user–defined parameters (section 4(d) *Local approximation evaluation*).

(a) Classical least squares fitting

Inferring the approximating polynomial vector field $\mathbf{F}(x,y) = \left(\mathbf{w}_{(x,y)}^t\mathbf{a},\ \mathbf{w}_{(x,y)}^t\mathbf{b} \right)$ reduces to computing the coefficient of $\mathbf{a}$ and $\mathbf{b}$ that minimize the approximation error. For least square methods, this error is formulated as the sum, for each point $\mathbf{p}_i$, of the squared distance between vectors $\mathbf{F}(\mathbf{p}_i)$ and $\mathbf{v}_i$, which can be written:

$$\begin{aligned} err(\mathbf{a},\mathbf{b}) &= \sum_{i=0}^{n} \|\mathbf{F}(\mathbf{p}_i) - \mathbf{v}_i\|^2 \\ &= \mathbf{a}^t\mathbf{S}\mathbf{a} + \mathbf{b}^t\mathbf{S}\mathbf{b} - 2\mathbf{a}^t\mathbf{S}_x - 2\mathbf{b}^t\mathbf{S}_y + S_{x,y} \end{aligned} \qquad (2)$$

where the following columns and matrices give a better description for the optimal solution:

$$\mathbf{S} := \sum_{i=0}^{n} \mathbf{w}_{(x_i,y_i)} \cdot \mathbf{w}_{(x_i,y_i)}^t \quad \in \mathbb{R}^{l\times l}$$

$$\mathbf{S}_x := \sum_{i=0}^{n} \left(\mathbf{v}_i^t \cdot \left[\begin{smallmatrix}1\\0\end{smallmatrix}\right]\right) \mathbf{w}_{(x_i,y_i)} \quad \in \mathbb{R}^{l}$$

$$\mathbf{S}_y := \sum_{i=0}^{n} \left(\mathbf{v}_i^t \cdot \left[\begin{smallmatrix}0\\1\end{smallmatrix}\right]\right) \mathbf{w}_{(x_i,y_i)} \quad \in \mathbb{R}^{l}$$

$$S_{x,y} := \sum_{i=0}^{n} \|\mathbf{v}_i\|^2 \quad \in \mathbb{R}.$$

With these definitions and using the normal equation, the critical point $(\mathbf{a},\mathbf{b})$ of the error function (2) is defined by:

$$\mathbf{S}\mathbf{a} = \mathbf{S}_x \qquad and \qquad \mathbf{S}\mathbf{b} = \mathbf{S}_y$$

Therefore, the coefficients of $\mathbf{a}$ and $\mathbf{b}$ that are the solution of our least square problem are obtained by solving two $l \times l$ systems of linear equations, which involves only the inversion of matrix $\mathbf{S}$: $\mathbf{a} = \mathbf{S}^{-1}\mathbf{S}_x$ and $\mathbf{b} = \mathbf{S}^{-1}\mathbf{S}_y$.

(b) Acceleration fitting

Some applications, like SPH simulations, also provide the derivative $\dot{\mathbf{v}}_i$ of the vector field at each point $\mathbf{p}_i \in \mathcal{P}$. We will call this field the *acceleration field*, since in these applications $\mathbf{v}_i$ usually represents the *velocity* of particle $\mathbf{p}_i$. This acceleration is usually given by the forces present at $\mathbf{p}_i$. In this case we can use the set acceleration vectors $\mathcal{A} = \{\dot{\mathbf{v}}_1, \dot{\mathbf{v}}_2, \ldots, \dot{\mathbf{v}}_n\}$ to complete the approximation of the velocity field.

Notice that the time varying acceleration vector at point $\mathbf{p}(t) = (x(t), y(t))$ should be approximated by the time derivative $\frac{d\mathbf{F}(\mathbf{p}(t))}{dt}$ of $\mathbf{F}$. Although we do not have expressions for $x(t)$ and $y(t)$, we do have the velocities at the points $\mathbf{p}_i \in \mathcal{P}$. The application of the chain rule thus defines $D\mathbf{F}(\mathbf{p}_i)\mathbf{v}_i$ as an estimate for the acceleration vector at $\mathbf{p}_i$, where $D\mathbf{F}(\mathbf{p}_i)$ is the Jacobian matrix of $\mathbf{F}$ at $\mathbf{p}_i$.

In order to improve the vector field approximation by the use of the set $\mathcal{A}$, we must add a new term to the least square problem (2). This term corresponds to the sum of the squared distance from the vector $D\mathbf{F}(\mathbf{p}_i)\mathbf{v}_i$ to $\dot{\mathbf{v}}_i$. Thus, the new minimization problem that balances the weight of the acceleration and the velocity approximation through a user–defined parameter μ is:

$$min_{\mathbf{a},\mathbf{b}} \left\{ \sum_{i=0}^{n} \|\mathbf{F}(\mathbf{p}_i) - \mathbf{v}_i\|^2 + \mu \sum_{i=0}^{n} \|D\mathbf{F}(\mathbf{p}_i)\mathbf{v}_i - \dot{\mathbf{v}}_i\|^2 \right\}$$

We can use again the column representation for P_d and Q_d to write the second term as:

$$\mathbf{a}^t\mathbf{Z}\mathbf{a} + \mathbf{b}^t\mathbf{Z}\mathbf{b} - 2\mathbf{a}^t\mathbf{Z}_x - 2\mathbf{b}^t\mathbf{Z}_y + Z_{x,y}$$

where the following columns and matrices give again a better description for the optimal solution:

$$\mathbf{D}_i := \left[\frac{\partial \mathbf{w}_{\mathbf{p}_i}}{\partial x}\ \frac{\partial \mathbf{w}_{\mathbf{p}_i}}{\partial y} \right] \quad \in \mathbb{R}^{l\times 2}$$

$$\mathbf{Z} := \sum_{i=0}^{n} \mathbf{D}_i\mathbf{v}_i\mathbf{v}_i^t\mathbf{D}_i^t \quad \in \mathbb{R}^{l\times l}$$

$$\mathbf{Z}_x := \sum_{i=0}^{n} \left(\dot{\mathbf{v}}_i^t \cdot \left[\begin{smallmatrix}1\\0\end{smallmatrix}\right]\right)\ \mathbf{D}_i\mathbf{v}_i \quad \in \mathbb{R}^{l}$$

$$\mathbf{Z}_y := \sum_{i=0}^{n} \left(\dot{\mathbf{v}}_i^t \cdot \left[\begin{smallmatrix}0\\1\end{smallmatrix}\right]\right)\ \mathbf{D}_i\mathbf{v}_i \quad \in \mathbb{R}^{l}$$

$$Z_{x,y} := \sum_{i=0}^{n} \|\dot{\mathbf{v}}_i\|^2 \quad \in \mathbb{R}.$$

As a consequence, the above acceleration fitting problem can be written:

$$\min_{\mathbf{a},\mathbf{b}} \Big\{ \mathbf{a}^t(\mathbf{S}+\mu\mathbf{Z})\mathbf{a} + \mathbf{b}^t(\mathbf{S}+\mu\mathbf{Z})\mathbf{b} \\ -2\mathbf{a}^t(\mathbf{S}_x+\mu\mathbf{Z}_x) - 2\mathbf{b}^t(\mathbf{S}_y+\mu\mathbf{Z}_y) + S_{x,y}+\mu Z_{x,y} \Big\}$$

To solve it, we need to find the critical point of the new error function. The optimal vectors $\mathbf{a}$ and $\mathbf{b}$ are thus obtained by solving the following two $l \times l$ systems of linear equations:

$$(\mathbf{S}+\mu\mathbf{Z})\mathbf{a} = (\mathbf{S}_x + \mu\mathbf{Z}_x) \ \ and\ \ (\mathbf{S}+\mu\mathbf{Z})\mathbf{b} = (\mathbf{S}_y + \mu\mathbf{Z}_y)$$

(c) Ridge regression

When the matrix $\mathbf{R} = (\mathbf{S} + \mu\mathbf{Z})$ doesn't have a maximal rank or is ill conditioned then the technique called *ridge regression* (RR) can be used to stabilize the linear system solutions [5] (see Figure 2). In our application, the RR technique modifies the optimization problem by adding two new terms depending on a diagonal matrix $\mathbf{\Delta} \in \mathbb{R}^{l\times l}$ and a constant scalar κ weighting the regression term:

$$\min_{\mathbf{a},\mathbf{b}} \Big\{ \mathbf{a}^t\mathbf{R}\mathbf{a} + \mathbf{b}^t\mathbf{R}\mathbf{b} - 2\mathbf{a}^t\mathbf{R}_x - 2\mathbf{b}^t\mathbf{R}_y + R_{x,y} \\ + \kappa\left(\mathbf{a}^t\mathbf{\Delta}\mathbf{a} + \mathbf{b}^t\mathbf{\Delta}\mathbf{b}\right) \Big\}$$

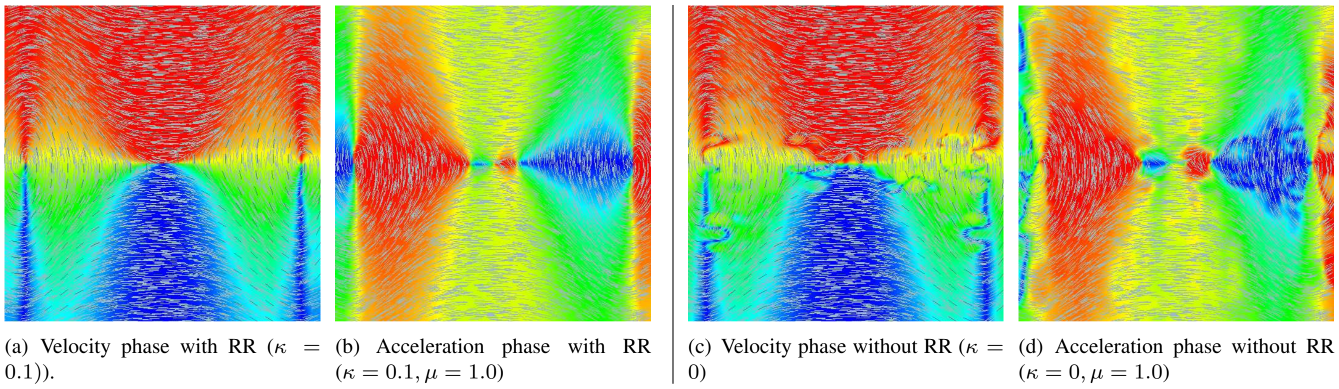

(a) Velocity phase with RR ($\kappa = 0.1$)). (b) Acceleration phase with RR ($\kappa = 0.1, \mu = 1.0$) (c) Velocity phase without RR ($\kappa = 0$) (d) Acceleration phase without RR ($\kappa = 0, \mu = 1.0$)

Figure 2: *The ridge regression (RR) improves the stability of the approximation.*

The solution of the ridge regression minimization problem is again obtained by solving the following system of linear equations:

$$(\mathbf{R} + \kappa\mathbf{\Delta})\mathbf{a} = \mathbf{R}_x \quad and \quad (\mathbf{R} + \kappa\mathbf{\Delta})\mathbf{b} = \mathbf{R}_y \qquad (3)$$

Instead of adopting an identity matrix for $\mathbf{\Delta}$, we preferred the one proposed by Tasdizen et al. [2]:

$$\mathbf{\Delta}_{\sigma\sigma} = \frac{i!j!}{(i+j)!}\left[\sum_{k,l\geq 0}^{k+l=i+j} \frac{(k+l)!}{k!l!} \sum_{m=1}^{q} x_m^{2k} y_m^{2l}\right],$$

where the indices $i, j \geq 0$ are deduced from index σ by $\sigma = j + \frac{(i+j+1)(i+j)}{2}$, with $i + j \leq d$. Such matrix has several interesting geometrical properties [2].

(d) Local approximation evaluation

Our local approximation scheme combines the least square fitting, the acceleration fitting and the ridge regression method. These three techniques are unified into the single square matrix inversion problem of equation (3). The user can set parameters μ and κ to use only part of the techniques. Observe that setting $\mu = 0$ and $\kappa = 0$ we have the classical least squares method. In particular, μ is set to zero when the acceleration field $\mathcal{A}$ is not available. Using $\kappa > 0$ we add the ridge regression term of the minimization.

5 From local to global vector field evaluation

The previous section detailed how we compute a local approximation $\mathbf{F}$ which fits to the field sample data $\mathcal{P}, \mathcal{V}$ and eventually $\mathcal{A}$. Because of its local nature, this approximation performs better on small sets of data. We thus use this approximation only on small *support regions* (section 5(a) *Adaptive domain subdivision*). To evaluate the approximate vector field $\mathbf{F}$ at a given point, we combine these local approximations using a multiresolution partition of unity (MPU) scheme (section 5(b) *Partition of unity*). This scheme guarantees a smooth behavior of $\mathbf{F}$, but requires that each region contains an approximation. This requirement can be satisfied using the proper multiresolution of the MPU scheme (section 5(b) *Partition of unity*). Figure 3 shows an example of how the polynomial degree and the multiresolution scheme influence on the field reconstruction.

(a) Adaptive domain subdivision

In order to benefit from the efficiency of the least–square method, we need to use it at the right level of detail. We define this level of detail through an adaptive quadtree decomposition of the vector field domain Ω: a cell of the quadtree is subdivided if it contains enough points for defining a polynomial of degree d (which has $\#P_d$ coefficients) and if it is not already of the maximal level (denoted by l_{max}) defined by the user. Figure 4(left) illustrates the domain subdivision determined by this quadtree structure, adapted to the input data (blue points).

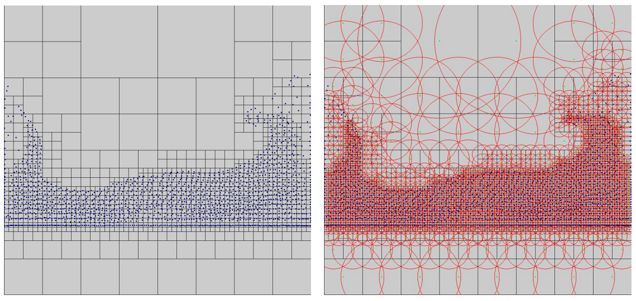


Figure 4: *Quadtree adaptated to the data and the leaf's supports.*

(b) Partition of unity

A *partition of the unity function* on $\Omega \subset \mathbb{R}^2$ is a set of positive functions $\varphi_i : \mathbb{R}^2 \to \mathbb{R}_+$ summing to 1 for each point of Ω: $\forall (x,y) \in \Omega, \quad \sum_i \varphi_i(x,y) \equiv 1$. These functions provide an optimal way of combining the different contributions of each local approximation: each of them can be weighted with a different φ_i, and their weighted sum will be a function $\mathbf{F}$ defined on the whole Ω. Moreover, since this sum is actually a convolution, $\mathbf{F}$ has the same regularity as the φ_i.

In practice, we use a multiresolution partition of unity, such as the one proposed by Ohtake et al. [10]. Each cell i of the quadtree defines a support region $supp(\varphi_i)$ for the φ_i, taken as a disk centered at the center $\mathbf{c}_i$ of the cell with radius $r_i = \frac{3}{4}$ of the diagonal of cell i. These support

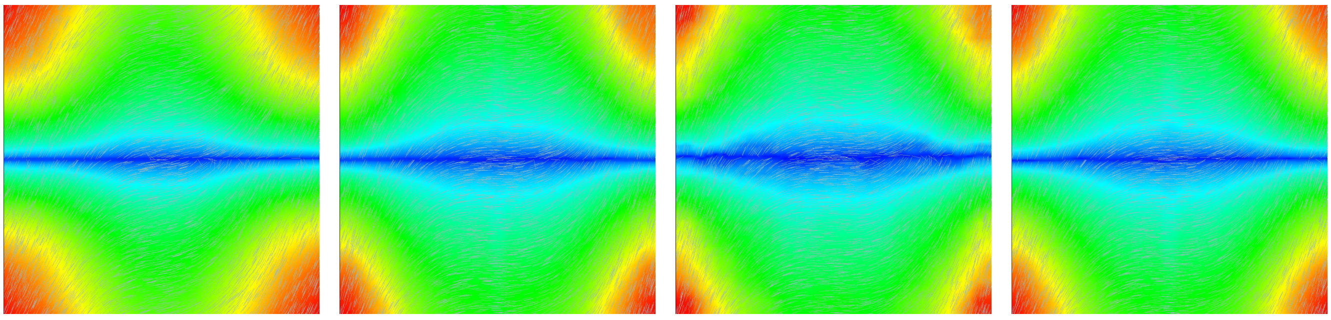

(a) $d = 2/l_{max} = 2$. (b) $d = 2/l_{max} = 5$. (c) $l_{max} = 5/d = 1$. (d) $l_{max} = 5/d = 3$.

Figure 3: *Multiresolution and polynomial degree effects on approximation.*

regions are illustrated on Figure 4(right). Then, we compute the local approximation $\mathbf{F}_i$ of the vector field $\mathcal{F}$ with the methods described at section 4 *Local vector field approximation*, but using only the points $\mathbf{p}_i$ and $\mathbf{v}_i$ belonging to $supp(\varphi_i)$. A global approximation for the vector field $\mathcal{F} : \Omega \to \mathbb{R}^2$ can then be deduced from the partition of unity by:

$$\mathcal{F}(x, y) \approx \mathbf{F}(x, y) \equiv \sum \varphi_i(x, y)\mathbf{F}_i(x, y). \tag{4}$$

(c) Kernel functions

The last step is to define the partition of unity for each cell i of the quadtree, respecting the support region and the constant sum restrictions. Since the number of neighbors j in the support varies from cell to cell, it is difficult to define directly a partition of unity which respects the constant sum restriction. The usual method defines φ_i from kernel functions k_i, based at the center $\mathbf{c}_i$ of the quadtree cell i. These kernels respect the support restriction, and the constant sum restriction is ensured by the following definition:

$$\varphi_i(x, y) = \frac{k_i(x, y)}{\sum_{j=1}^{n} k_j(x, y)} \tag{5}$$

There are several examples of kernel functions with this type of compact support and whose range is contained in the interval $[0, 1]$. We mainly used the *poly6* kernel [4]:

$$k_i(x, y) = \max\left(0\ ,\ \frac{4}{\pi r_i^8}(r_i^2 - \|(x, y) - c_i\|^2)^3\right)$$

(d) Global approximation evaluation

We have now all the elements to compute a vector field $\mathbf{F} : \Omega \to \mathbb{R}^2$ that approximates the vector field $\mathcal{F}$ from where the data was sampled. To evaluate $\mathbf{F}$ at a point $\mathbf{p} \in \Omega$, we traverse the quadtree, enumerating the n_f leaf cells whose support region contains $\mathbf{p}$. After that, we compute the $P(\mathbf{p})$ and $Q(\mathbf{p})$ coordinate function of $\mathbf{F}(\mathbf{p})$ using equations (4) and (5).

However, in order to solve the two $l \times l$ linear systems of the local approximation (equation (3)), each cell must contain at least $l = \#P_p$ sampled points in its support region. For example, using a polynomial approximation of degree $d = 2$, we need $l = 6$ points inside each support region. We propose the following strategy to work when this number is not reached for the support of cell i: we generate random points uniformly inside the support region of cell i. We attach to these points the vector obtained by evaluating the polynomial approximation $\mathbf{F}_f$ of the father f of i.

6 Application to derivatives evaluation

In this section, we present how to apply our approximated velocity field to the computation of integral curves and to estimate Jacobian matrices and acceleration vectors.

Integral curves. Our method allows computing integral curves on Ω using the global approximation for the velocity field $\mathbf{F} : \Omega \to \mathbb{R}^2$. Given an initial condition $\mathbf{p}_0 \in \Omega$, the *integral curve* at $\mathbf{p}_0$ is the function $c_{\mathbf{p}_0} : \mathbb{R} \to \mathbb{R}^2,\quad t \mapsto c_{\mathbf{p}_0}(t)$ that satisfies:

$$c_{\mathbf{p}_0}(0) = \mathbf{p}_0; \qquad \frac{dc_{\mathbf{p}_0}}{dt}(t) = \mathbf{F}(c_{\mathbf{p}_0}(t)).$$

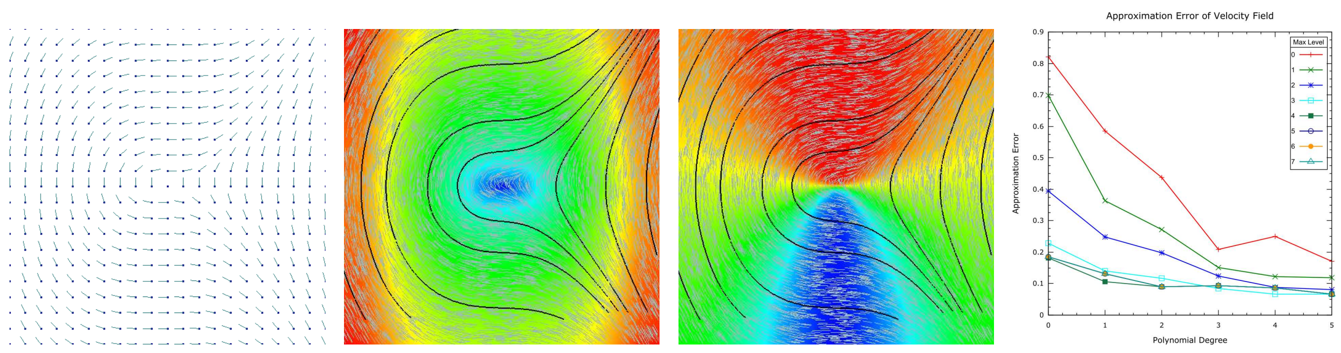
We can compute these integral lines using an Euler method on the global evaluation for $\mathbf{F}$. The last picture of Figure 1 shows examples of integral curves using several initial conditions.

Jacobian matrix evaluation. Using the expression of the velocity field's global approximation described in (4), we can also compute an estimative for the Jacobian matrix of $\mathbf{F}$ at a given point $\mathbf{p}$. This requires computing the partial derivatives of its coordinate functions. The expressions for the coordinate functions are given by the following formula:

$$\frac{\partial P}{\partial x}(\mathbf{p}) = \sum_{i=1}^{n_f} \varphi_i(\mathbf{p})\frac{\partial P_{d,i}}{\partial x}(\mathbf{p}) + \sum_{i=1}^{n_f} \frac{\partial \varphi_i}{\partial x}(\mathbf{p})P_{d,i}(\mathbf{p})$$

$$\frac{\partial P}{\partial y}(\mathbf{p}) = \sum_{i=1}^{n_f} \varphi_i(\mathbf{p})\frac{\partial P_{d,i}}{\partial y}(\mathbf{p}) + \sum_{i=1}^{n_f} \frac{\partial \varphi_i}{\partial y}(\mathbf{p})P_{d,i}(\mathbf{p})$$

Acceleration field evaluation. We can apply the above formulas to obtain an approximation of the acceleration field: the acceleration vector at point $\mathbf{p}$ is given by $D\mathbf{F}(\mathbf{p})\mathbf{v}$, where $D\mathbf{F}(\mathbf{p})$ is the Jacobian matrix of $\mathbf{F}$ at $\mathbf{p}$, and $\mathbf{v} = \mathbf{F}(\mathbf{p})$ is the velocity vector. This approximation serves not only for computational fluid application, but also for visualizing properties of the resulted reconstruction.



(a) Original sampled data.

(b) Magnitude map of the approximated field.

(c) Phase map of the approximated field.

(d) Error graph of the synthetic example.

Figure 5: *Synthetic field example $\mathcal{F} = (y, x^2)$, with some integral curves.*

7 Results

In this section we use the following convention for the colors on the results figures background. For magnitude maps, the colors vary from blue to red representing an scale of the magnitude from low to high values. For phase maps, the colors represent the cosine of the phase (which is the angle between the vector and the abscissa axis). Again, we use a color palette that varies from blue to red, representing the variation of the cosine from -1 to 1.

We also use the following convention for the approximation error graphs. The abscissa represents the degree of the polynomial, the ordinate represents global approximation error. In each graph illustrating the errors, we draw eight curves: one for each value of the maximal level l_{max}. We choose to vary l_{max} from 0 to 7.

To measure the quality of the approximation, we use the following error formula:

$$error = \frac{1}{n}\sum_{i=1}^{n}\|\mathbf{F}(\mathbf{p}_i) - \mathbf{v}_i\| \Bigg/ \frac{1}{n}\sum_{i=1}^{n}\|\mathbf{v}_i\| \qquad (6)$$

This formula computes the quotient of the mean approximation distance error and the mean velocity norm using all samples. We illustrate the power of our method by the use of four examples.

Synthetic field. The first example illustrates the reconstruction of a set of points sampled from the velocity field $\mathcal{F}(x, y) = (y, x^2)$ on the region $[-2, 2] \times [-2, 2]$. Figure 5(a) shows the 441 sampled points with their corresponding velocity vectors. Figures 5(b)(c) show the visualization of some reconstructed velocity vectors and integral curves. At the background we see on image (b) the velocity magnitude map and on (c) the cosine of the velocity phase map. For that reconstruction, we use the following parameters: $d = 2$, $\mu = 0$, $\kappa = 0.1$ and $l_{max} = 6$.

Figure 5(d) shows the approximation error of the reconstruction using the formula (6). Observing this graph, we conclude that we get better approximations either when we increase the degree or when we increase the value of l_{max}.

Stable fluids. In figure 1 we provide an example of a velocity field reconstruction obtained from 4096 samples of an Eulerian grid-based fluid simulation [12]. From left to right, the first image shows a discretized velocity field of a smoke flow, the second illustrates the field reconstruction obtained by the method and its magnitude map. The third one shows the reconstructed field and its phase map at the background. Finally, the last image displays some integral curves using the reconstructed field. For that reconstruction, we use the following parameters: $d = 2$, $\mu = 0$, $\kappa = 0.1$ and $l_{max} = 6$. Figure 7 shows a graph of the approximation error computed on the samples.

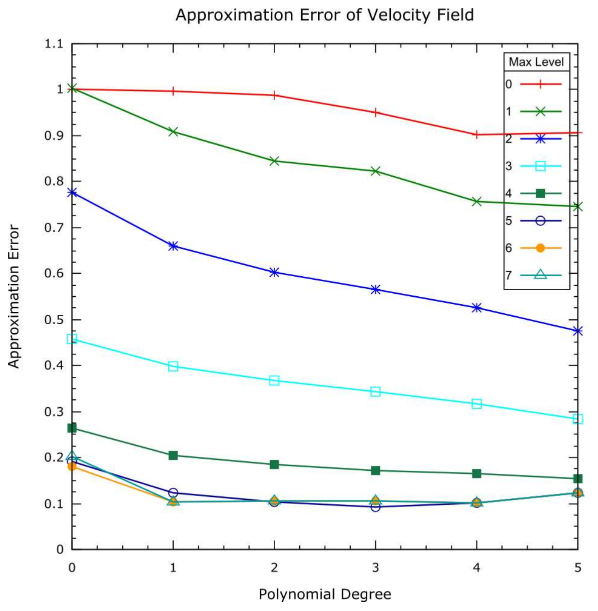
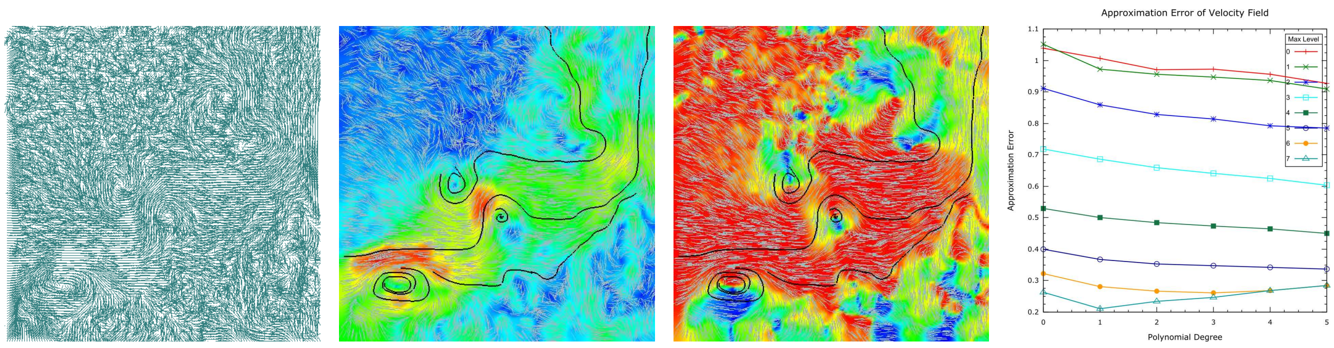


Figure 7: *Stable fluid approximation error.*

Particle image velocimetry. An important application of our method is on the reconstruction of sampled points of vector field acquired from a PIV device. Figure 6(a) shows the input data with 15607 points. This sampled velocity field corresponds to a flow of a gas that is continuously injected



(a) Original data. (b) Magnitude map of the velocity approximation. (c) Phase map of the velocity approximation. (d) Global error graph.

Figure 6: *Particle image velocimetry example, with some integral curves.*

horizontally on the bottom left corner. This gas flows on the domain from left to right until it meets an wall, represented on the image by its right edge.

One can visualize some reconstructed velocity vectors and integral curves in figures 6(b)(c). Again, at the background we see on image (b) the velocity magnitude map and on image (c) the cosine of the velocity phase map. For this example we use $d = 2$, $\mu = 0$, $\kappa = 0.1$ and $l_{max} = 6$. Figure 6 shows the approximation error. In this case the error is higher than the previous examples because the data is very noisy in the top left and bottom right corners.

Smooth particle hydrodynamics. In the SPH application, the acceleration vector is available at each sampled point. In that case, we can use our acceleration fitting method for local approximations. The initial condition for the 2D SPH simulation is a rested fluid box, dropped at the bottom center of a rectangular container. The data input on this example corresponds to 1800 fluid particles at a given time of the simulation, together with the following attributes: position, velocity and acceleration.

Figure 8(a) shows the input points and the corresponding velocities after of the impact against the vertical walls. Figures 8(b),(c) show the reconstructed velocity vectors and integral curves together with the velocity magnitude and cosine of the phase maps. Again this example is very noisy. Finally, Figure 8(d) shows the approximation error graph.

This example shows that there exists an optimal level, since even we continue to increase the maximum level we can't improve the approximation. The reason is that the support region of a high level node may not contain sufficient number of point to make a good local approximation.

8 Conclusions and future works

This work proposed a novel multiresolution scheme for velocity field reconstruction from sparse sampled points. This new scheme combines three important techniques, *least squares fitting*, *ridge regression* and *partition of unity*, to produce a global approximation of the velocity field. The method could be used on samples from very different sources. The local approximation procedure is very flexible, since it unifies several methods and control their functionality by the use of parameters (d,μ, and κ). The global approximation is obtained by the use of a *partition of unity*. Again the global approximation shows to be very malleable, since the users not only have several options for the kernel functions, but also can choose the maximum level of the *QuadTree* and the local error control threshold to control the reconstruction result.

The authors plan to extend this work in three main directions. One is to generalize it to 3D velocity field reconstruction. Another direction is to produce a velocity field reconstruction scheme that is conservative. And the other is to use other subdivision schemes, like *binary space partitions*, to improve the approximation.

Acknowledgments

The authors are members of the Matmidia laboratory, which is sponsored by FINEP, PETROBRAS, CNPq, and FAPERJ. H. Lopes is partially supported by a CNPq and FAPERJ Primeiro Projeto research grants.

References

[1] I. Babuska and J. M. Melenk. The partition of unity method. *International Journal of Numerical Methods in Engineering*, 40:727–758, 1997.

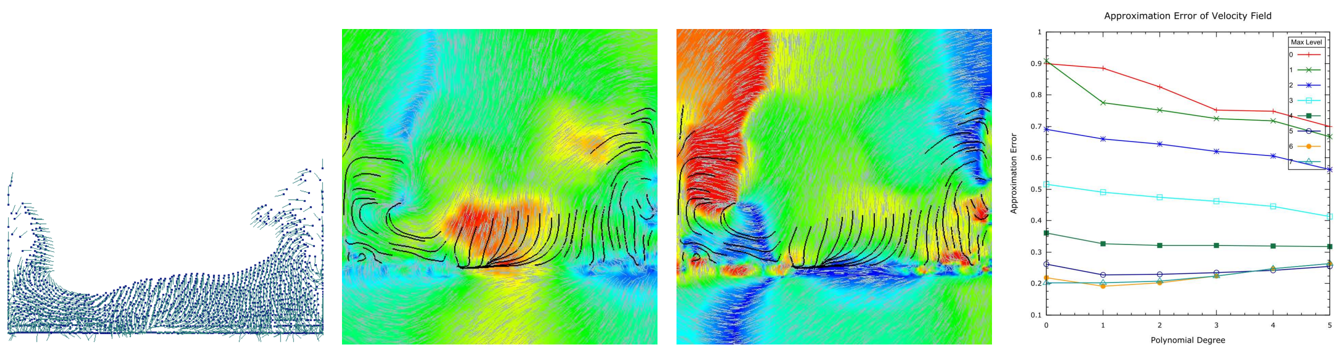
[2] T. Tasdizen, J. P. Tarel and D. B. Cooper. Algebraic curves that work better. In *Computer Vision and Pattern Recognition*, IEEE, pages 35–41, 1999.

[3] T. Tasdizen, J.-P. Tarel and D. Cooper. Improving the stability of algebraic curves for applications. *Transactions on Image Processing*, 9(3):405–416. 2000.

[4] M. Desbrun and M.-P. Gascuel. Smoothed particles : A new paradigm for animating highly deformable bodies. In *Comp. Animation and Simulation*, pages 61–76, 1996.

[5] A. Hoerl and R. Kennard. Ridge regression: Biased estimation for nonorthogonal problems. *Technometrics*, 12(3):55–67, 1970.

[6] D. Keren. Topologically faithful fitting of simple closed curves. *IEEE PAMI*, 26(1):118–123, 2004.



(a) Original sampled field. (b) Magnitude map of the velocity approximation. (c) Phase map of the velocity approximation. (d) Approximation error graph.

Figure 8: *SPH simulation example, with some integral curves.*

[7] P. Lancaster and K. Salkauskas. *Curve and Surface Fitting: An Introduction*. Academic Press, 1986.

[8] M. Muller, D. Charypar and M. Gross. Particle-based fluid simulation for interactive applications. In *Symposium on Computer animation*, pages 154–159. 2003.

[9] M. Muller, S. Schirm and M. Teschner. Interactive blood simulation for virtual surgery based on smoothed particle hydrodynamics. *Technological Health Care*, 12(1):25–31, 2004.

[10] Y. Ohtake, A. Belyaev, M. Alexa, G. Turk and H.-P. Seidel. Multi-level partition of unity implicits. *ACM Transactions on Graphics*, 22(3):463–470, 2003.

[11] M. Raffel, C. Willert and J. Kompenhans. *Particle Image Velocimetry: A Practical Guide*. Springer, 2002.

[12] J. Stam. Stable fluids. In *Siggraph*, pages 121–128, 1999.

[13] M. Stanislas, J. Kompenhans and J. Westerweel. *Particle Image Velocimetry*. Springer, 2000.

[14] G. Taubin. Estimation of planar curves, surfaces, and nonplanar space curves defined by implicit equations *IEEE PAMI*, 13(11):1115–1138, 1991.

[15] R. Franke and G. Nielson. Smooth interpolation of large sets of scattered data. *International Journal of Numerical Methods in Engineering*, 15:1691–1704, 1980.

[16] A. K. Jain. *Fundamentals of Digital Image Processing*. Prentice Hall, 1988.

[17] J. J. Monaghan. Smoothed particle hydrodynamics. *Reports on Progress in Physics*, 68:1703–1759, 2005.

[18] H. Pottmann and T. Randrup. Rotational and helical surface approximation for reverse engineering. *Computing*, 60(4):307–302, 1998.

[19] M. Alexa, J. Behr, D. Cohen-Or, S. Fleishman, D. Levin and C. T. Silva. Computing and rendering point set surfaces. *IEEE Transactions on Visualization and Computer Graphics*, 9(1):3–15, 2003.