Published in Image Processing On Line on 2020–06–28.
Submitted on 2019–10–02, accepted on 2020–01–24.
ISSN 2105–1232 © 2020 IPOL & the authors CC–BY–NC–SA
This article is available online with supplementary materials,
software, datasets and online demo at
https://doi.org/10.5201/ipol.2020.281

On Anisotropic Optical Flow Inpainting Algorithms

Lara Raad[1], Maria Oliver[2], Coloma Ballester[1], Gloria Haro[1], Enric Meinhardt[3]

[1] UPF, Barcelona, Spain
({lara.raad,coloma.ballester,gloria.haro}@upf.edu)
[2] Institute of Technology, U. Grenoble, France
(Maria.Oliver-Parera@gipsa-lab.grenoble-inp.fr)
[3] CMLA,ENS Paris-Saclay, France
(enric.meinhardt@cmla.ens-cachan.fr)

Abstract

This work describes two anisotropic optical flow inpainting algorithms. The first one recovers the missing flow values using the Absolutely Minimizing Lipschitz Extension partial differential equation (also called infinity Laplacian equation) and the second one uses the Laplace partial differential equation, both defined on a Riemmanian manifold. The Riemannian manifold is defined by endowing the plane domain with an appropriate metric depending on the reference video frame. A detailed analysis of both approaches is provided and their results are compared on three different applications: flow densification, occlusion inpainting and large hole inpainting.

Source Code

The C and Octave/Matlab source codes implementing the algorithms described in the paper, and the online demo, are accessible at the associated web page[1].

Keywords: optical flow inpainting; absolutely minimizing Lipschitz extension; Laplace-Beltrami

1 Introduction

Optical flow inpainting is a pervasive problem in many tasks in computer vision ranging from semantic video analysis and video editing to optical flow estimation in occlusion and disocclusion regions. Occluded and nonoccluded areas represent one of the major difficulties in optical flow estimation due to the lack of correspondences with the next and previous frame respectively. A common solution is to inpaint or fill in the optical flow in such regions. In this work the problem of filling in an incomplete optical flow is addressed using anisotropic interpolation in Riemannian manifolds, as proposed in [18]. The interpolation is solved on a Riemannian manifold in order to take advantage of the geometric information given by the video frames: Given a video and an incomplete motion field, each 2D frame domain is endowed with a Riemannian metric based on the video values. Then, the

[1] https://doi.org/10.5201/ipol.2020.281

missing optical flow is recovered by solving an appropriate Partial Differential Equation, such as the Absolutely Minimizing Lipschitz Extension (AMLE) or the Laplace equations, on a 2D Riemannian manifold. The inpainting process is then stated as a boundary value problem given the known values on the boundary of the interpolation domain, which may include isolated points. The two coordinates of the optical flow are separately reconstructed. Some authors have approached the problem with different algorithms (see, e.g., [9, 5]).

The AMLE equation, $\Delta_\infty u = 0$, where $\Delta_\infty u$ is also called the infinity Laplacian, was introduced by G. Aronsson in [2, 3] and uniqueness of viscosity solutions was proved in [12] (see also [4] for a review). The AMLE allows to interpolate the missing values of a function given on curves and disconnected points. It appeared as one of the interpolating operators satisfying a set of suitable axioms in [8]. This axiomatic approach was extended in [19, 7] to interpolate data given on a set of curves on a surface in $\mathbb{R}^3$. The AMLE and the *biased* AMLE on a manifold were applied in [13] for depth interpolation in images or videos where, due to acquisition failure, large regions of incomplete depth information often appear. The classical AMLE has been also used in [1] for the interpolation of digital elevation models.

The Laplace-Beltrami operator can also be used for anisotropic interpolation in Riemannian manifolds and will be used in this paper to recover the missing regions of the optical flow. The Laplace-Beltrami operator is a generalization of the Laplacian to Riemannian manifolds such as surfaces or as the Euclidean space $\mathbb{R}^N$ endowed with a non-Euclidean metric. The anisotropic diffusion produced by the Laplace-Beltrami operator has been extensively used in many areas such as image processing or shape analysis.

In this work a detailed analysis and description of [18] and a variant of it using the Laplace-Beltrami operator are provided. The rest of the paper is organized as follows. Section 2 explains the AMLE in a manifold model, together with its discretization using the finite graph formalism. It includes an analysis of different metric choices, discretization resolution and parameters boosting the discrete model accuracy. Section 3 describes the Laplace-Beltrami operator used to interpolate the optical flow. It includes an analysis of different weights and a graph formalism based approach reducing the problem to a simple linear system. A detailed description of the parameters and the algorithms are provided in Section 4 and Section 5 respectively. Section 6 illustrates the behaviour of both algorithms with experimental results. Finally, Section 7 concludes the paper.

2 The AMLE in a Manifold Model

Given a video $I(\mathbf{x}, t)$ defined on $\Omega \times \{1, \dots, T\}$, where $\Omega \subset \mathbb{R}^2$ denotes the image frame domain and $\{1, \dots, T\}$ is the set of discrete times, let $\mathbf{v}(\mathbf{x}, t) = (v_1(\mathbf{x}, t), v_2(\mathbf{x}, t))$ be the optical flow of the video I, i.e., the apparent motion between a pixel $\mathbf{x} \in \Omega$ at time t and the corresponding one at time $t+1$.

At time t, $\mathbf{v}(\mathbf{x}, t)$ is assumed to be unknown on an open region $\Omega_0(t) \subset \Omega$ whose boundary, denoted by $\partial\Omega_0(t)$, consists of a finite union of smooth curves and possibly isolated points. In order to interpolate $\mathbf{v}$ in $\Omega_0(t)$ taking into account the objects in the video, the whole domain Ω is endowed, at each time t, with a metric $g(t)$. Let $\mathcal{M}(t) = (\Omega, g(t))$ be the corresponding Riemannian manifold. The motion field $\mathbf{v}$ in $\Omega_0(t)$ is completed, at each time $t \in \{1, \dots T-1\}$, with $\mathbf{u} = (u_1, u_2)$ such that u_1 and u_2 are solutions, respectively, of the geodesic AMLE equation

$$\begin{cases} \Delta_{\infty,g} u_c = 0 & \text{in } \Omega_0(t), \\ u_c = v_c, & \text{in } \partial\Omega_0(t), \end{cases} \tag{1}$$

for $c = 1, 2$, respectively. Newmann boundary conditions are used on $\partial\Omega$. The operator $\Delta_{\infty,g} u_c$ is defined as

$$\Delta_{\infty,g} u_c := D^2_{\mathcal{M}} u_c \left(\frac{\nabla_{\mathcal{M}} u_c}{|\nabla_{\mathcal{M}} u_c|}, \frac{\nabla_{\mathcal{M}} u_c}{|\nabla_{\mathcal{M}} u_c|} \right), \tag{2}$$

where $\nabla_{\mathcal{M}} u_c$ and $D^2_{\mathcal{M}} u_c$ denote, respectively, the gradient and the Hessian of u_c on the manifold. To simplify the notation, the dependence on t of g and $\mathcal{M}$ was omitted and will be omitted throughout the document; the subindex c (associated to the optical flow coordinate) will also be dropped in what follows if no confusion may be caused. Using a coordinate system and the Einstein's convention, the PDE in (1) can be written as

$$\left(\frac{\partial^2 u}{\partial x^i \partial x^j} - \Gamma^k_{ij}(\mathbf{x}) \frac{\partial u}{\partial x^k} \right) g^{ir}(\mathbf{x}) g^{js}(x) \frac{\partial u}{\partial x^r} \frac{\partial u}{\partial x^s} = 0 \qquad \text{in } \Omega_0, \tag{3}$$

where $\Gamma^k_{ij}(x)$ denote the Christoffel symbols in that coordinate system. When $\mathcal{M} = \mathbb{R}^n$ and g is the Euclidean metric, the PDE (1) is called the infinity Laplace equation and the operator in (2) is known as the infinity Laplacian. For our application, the manifold $\mathcal{M}$ is a rectangle of $\mathbb{R}^2$ endowed with a Riemannian metric g. The metric g is defined in this paper based on the local geometry and texture content of video frames and several definitions are tested. For instance, g can be given by affine covariant structure tensors [10] or taking into account spatial distances and photometric similarities as detailed in Section 2.2.

2.1 The Discrete Model

The AMLE equation is solved on the manifold with a numerical algorithm which is based on the eikonal operators for nonlinear elliptic PDEs on a finite graph and which was proposed by Manfredi et al. [17, 16]. In particular, the discrete grid of $\Omega \subset \mathbb{R}^2$ is considered as the nodes of a finite weighted graph G. As usual, we consider a grid size $h = 1$ in the horizontal and vertical directions. We will denote by V the set of vertices of $G = (V, \mathcal{E}, \omega)$, where $\mathcal{E} \subset V \times V$ denotes the set of edges between pairs of nodes, and $\omega : \mathcal{E} \to \mathbb{R}$ denotes the weight map defined on the edges. These weights on the edges depend on the given metric g and are described in Section 2.2.

Given a point $\mathbf{x}$ on the grid, let $\mathcal{N}(\mathbf{x})$ be a local neighborhood of $\mathbf{x}$. Following [16], the positive and negative eikonal operators are given, respectively, by

$$\|\nabla u(\mathbf{x})\|^+_{\mathcal{M}} = \max_{\mathbf{y} \in \mathcal{N}(\mathbf{x})} \frac{u(\mathbf{y}) - u(\mathbf{x})}{d^g(\mathbf{x}, \mathbf{y})}, \qquad \|\nabla u(\mathbf{x})\|^-_{\mathcal{M}} = \min_{\mathbf{z} \in \mathcal{N}(\mathbf{x})} \frac{u(\mathbf{z}) - u(\mathbf{x})}{d^g(\mathbf{x}, \mathbf{z})}, \tag{4}$$

where $d^g(\cdot, \cdot)$ denotes the distance according to the metric g between two points. Then, the discrete infinity Laplacian corresponds to

$$\Delta_{\infty, g} u(\mathbf{x}) = \frac{\|\nabla u(\mathbf{x})\|^+_{\mathcal{M}} + \|\nabla u(\mathbf{x})\|^-_{\mathcal{M}}}{2}. \tag{5}$$

This yields the following iterative discrete scheme used to solve (1)

$$u^{k+1}(\mathbf{x}) = \frac{d^g(\mathbf{x}, \mathbf{z}) u^k(\mathbf{y}) + d^g(\mathbf{x}, \mathbf{y}) u^k(\mathbf{z})}{d^g(\mathbf{x}, \mathbf{z}) + d^g(\mathbf{x}, \mathbf{y})}, \tag{6}$$

where $\mathbf{y}$ and $\mathbf{z}$ are the pixels providing the maximum and minimun in (4). As proved in [17] the solution of the numerical scheme (6) converges to the solution of (1) when the local spatial resolution dx and the local directional resolution $d\theta$ in the neighborhood $\mathcal{N}(\mathbf{x})$ tend to 0. For $\mathbf{x}$ on the grid, let $\mathbf{x_1}, \ldots, \mathbf{x_N} \in \mathcal{N}(\mathbf{x})$ denote its neighbors. The local spatial resolution is defined as

$$dx = \max_{i=1}^{N} |\mathbf{p}_i|, \tag{7}$$

where $\mathbf{p}_i = \mathbf{x} - \mathbf{x_i}$ is the direction vector. The local directional resolution is defined as

$$d\theta = \max_{\mathbf{p} \in S^N} \min_{i=1}^{N} |\mathbf{p} - \hat{\mathbf{p}}_i|, \tag{8}$$

where S^N is the set of direction vectors $\{\mathbf{p}_i\}_{i=1}^N$ and $\hat{\mathbf{p}}_i = \mathbf{p}_i/|\mathbf{p}_i|$. The effect of these resolutions will be analyzed in Section 2.4. In [18], this scheme is applied for $\mathbf{x} \in \Omega_0$ keeping the known values of $v_1(\mathbf{x})$, respectively $v_2(\mathbf{x})$, on the known region $\Omega \setminus \Omega_0$ for all k, and initializing $u^{k=0}(\mathbf{x}) = 0$ in Ω_0.

2.2 An Approximation of d^g

Given any two points $\mathbf{x}$ and $\mathbf{y}$ on the grid, the geodesic distance $d^g(\mathbf{x}, \mathbf{y})$ is defined as

$$d^g(\mathbf{x}, \mathbf{y}) = \inf_{\gamma}\{ \text{length}^g(\gamma) : \gamma \text{ is a curve on the grid joining } \mathbf{x} \text{ and } \mathbf{y}\}. \tag{9}$$

This distance can be computed using Dijkstra's algorithm, given the distance between adjacent nodes on the grid graph of the image. In practice, for any pair of pixels $\mathbf{x}, \mathbf{y}$ on a neighborhood $\mathcal{N}$ and on the grid, we approximate the geodesic distance $d^g(\mathbf{x}, \mathbf{y})$ in (9) by $d(\mathbf{x}, \mathbf{y})$, where $d(\mathbf{x}, \mathbf{y})$ is defined by one of the following possibilities:

$$d_1(\mathbf{x}, \mathbf{y}) = \sqrt{(1-\lambda)\|I(\mathbf{x}, t) - I(\mathbf{y}, t)\|^2 + \lambda\|\mathbf{x} - \mathbf{y}\|^2}, \tag{10}$$
$$d_2(\mathbf{x}, \mathbf{y}) = (1-\lambda)\|I(\mathbf{x}, t) - I(\mathbf{y}, t)\| + \lambda\|\mathbf{x} - \mathbf{y}\|, \tag{11}$$
$$d_3(\mathbf{x}, \mathbf{y}) = (1-\lambda)\|I(\mathbf{x}, t) - I(\mathbf{y}, t)\|^2 + \lambda\|\mathbf{x} - \mathbf{y}\|^2, \tag{12}$$
$$d_4(\mathbf{x}, \mathbf{y}) = (1-\lambda)\|\mathcal{P}(\mathbf{x}, t) - \mathcal{P}(\mathbf{y}, t)\|^2 + \lambda\|\mathbf{x} - \mathbf{y}\|^2, \tag{13}$$

where $\lambda \in [0, 1]$ and $\mathcal{P}(\mathbf{x}, t)$ denotes a patch of I of size $s \times s$ pixels centered at $(\mathbf{x}, t)$. Notice that the definition in (12) is a semimetric, i.e., it does not satisfy the triangle inequality, and that (13) is a generalization of (12) using patches. From now on, the discrete scheme in (6) will be referred in terms of the weights $\omega_i(\mathbf{x}, \mathbf{y}) = 1/d_i(\mathbf{x}, \mathbf{y}),\ i \in \{1, \dots, 4\}$, which yields the following discrete scheme

$$u^{k+1}(\mathbf{x}) = \frac{1}{\omega_i(\mathbf{x}, \mathbf{z}) + \omega_i(\mathbf{x}, \mathbf{y})} \left(\omega_i(\mathbf{x}, \mathbf{y})u^k(\mathbf{y}) + \omega_i(\mathbf{x}, \mathbf{z})u^k(\mathbf{z})\right). \tag{14}$$

To experimentally analyze the behavior of the interpolation method with respect to the different metric g (thus, on the different weights' choice) we present in Figures 1 and 2 the inpainting results using the weights $\omega_i(\mathbf{x}, \mathbf{y}),\ i \in \{1, \dots, 4\}$, based on the definitions (10), (11), (12) and (13). Notice that $\omega_3(\mathbf{x}, \mathbf{y})$ (given by the semi-metric defined in (12)) produces slightly better results. Indeed, when part of an edge is subjective or weak (i.e., the two regions separated by the edge are similar) the other weights do not completely penalize the propagation of the optical flow from one region to another; $\omega_3(\mathbf{x}, \mathbf{y})$ does a better job as observed in Figure 1. In Figure 2 one can notice that the edges of the different moving objects are sharper when $\omega_3(\mathbf{x}, \mathbf{y})$ is used. For all the results in Figure 1 and Figure 2 the End-Point Error (EPE), with respect to the ground truth flow $\bar{\mathbf{v}} = (\bar{v}_1, \bar{v}_2)$, is indicated. The formula used to calculate the EPE is

$$E = \frac{1}{|\Omega_0|} \sum_{\mathbf{x} \in \Omega_0} \sqrt{(\bar{v}_1(\mathbf{x}) - u_1(\mathbf{x}))^2 + (\bar{v}_2(\mathbf{x}) - u_2(\mathbf{x}))^2}. \tag{15}$$

The influence of the parameter λ is shown in Figure 3, where three inpainting examples are considered. Each curve of the last row plots the interpolation error for each weight and for different values of λ. These curves show that there is not a unique optimal value of λ but rather an optimal one for each image and weight $\omega_i(\mathbf{x}, \mathbf{y})$. One can observe different behaviours depending on the content of the reference frame, which defines the metric, and the content of the optical flow. In some cases the value of λ might almost not affect the error for a given weight (column one). In others the optimal value is $\lambda = 1$, i.e. no metric is used (column three).

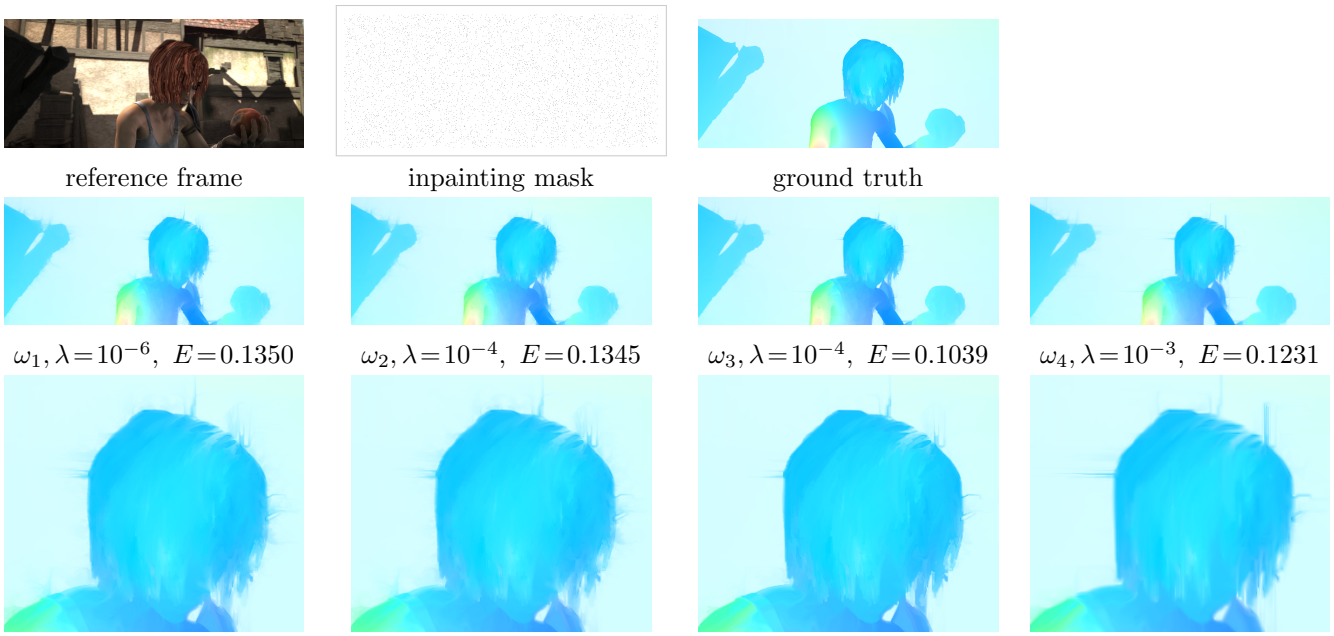

Figure 1: AMLE interpolation. Influence of the metric. First row from left to right, the guiding image, the sparse inpainting mask with 1% points where white pixels denote unknown optical flow values, and the ground truth flow. Second row shows four inpainting results corresponding from left to right to the weights $\omega_1(\mathbf{x}, \mathbf{y})$, $\omega_2(\mathbf{x}, \mathbf{y})$, $\omega_3(\mathbf{x}, \mathbf{y})$ and $\omega_4(\mathbf{x}, \mathbf{y})$. The interpolation error and the value of λ are indicated for each result. The last row shows a detail of the results of the second row.

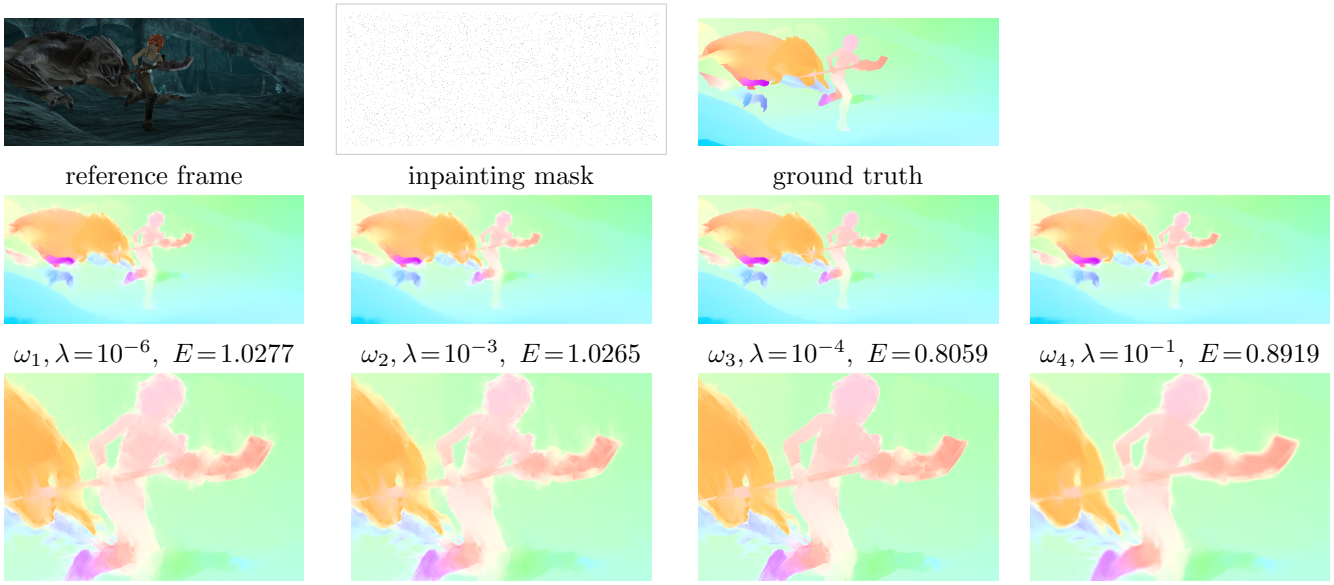

Figure 2: AMLE interpolation. Influence of the metric. First row from left to right, the guiding image, the sparse inpainting mask with 1% points where white pixels denote unknown optical flow values, and the ground truth flow. Second row shows four inpainting results corresponding from left to right to the weights $\omega_1(\mathbf{x}, \mathbf{y})$, $\omega_2(\mathbf{x}, \mathbf{y})$, $\omega_3(\mathbf{x}, \mathbf{y})$ and $\omega_4(\mathbf{x}, \mathbf{y})$. The interpolation error and the value of λ are indicated for each result. The last row shows a detail of the results of the second row.

2.3 Influence of the Image used to Compute the Image-Based Metric

As mentioned before, the metric g is computed based on local geometry and texture content given by an underlying image called the guiding image. In this section the influence of this guiding image is analyzed. To this goal two inpainting results are shown in Figures 4 and 5. Both of them show

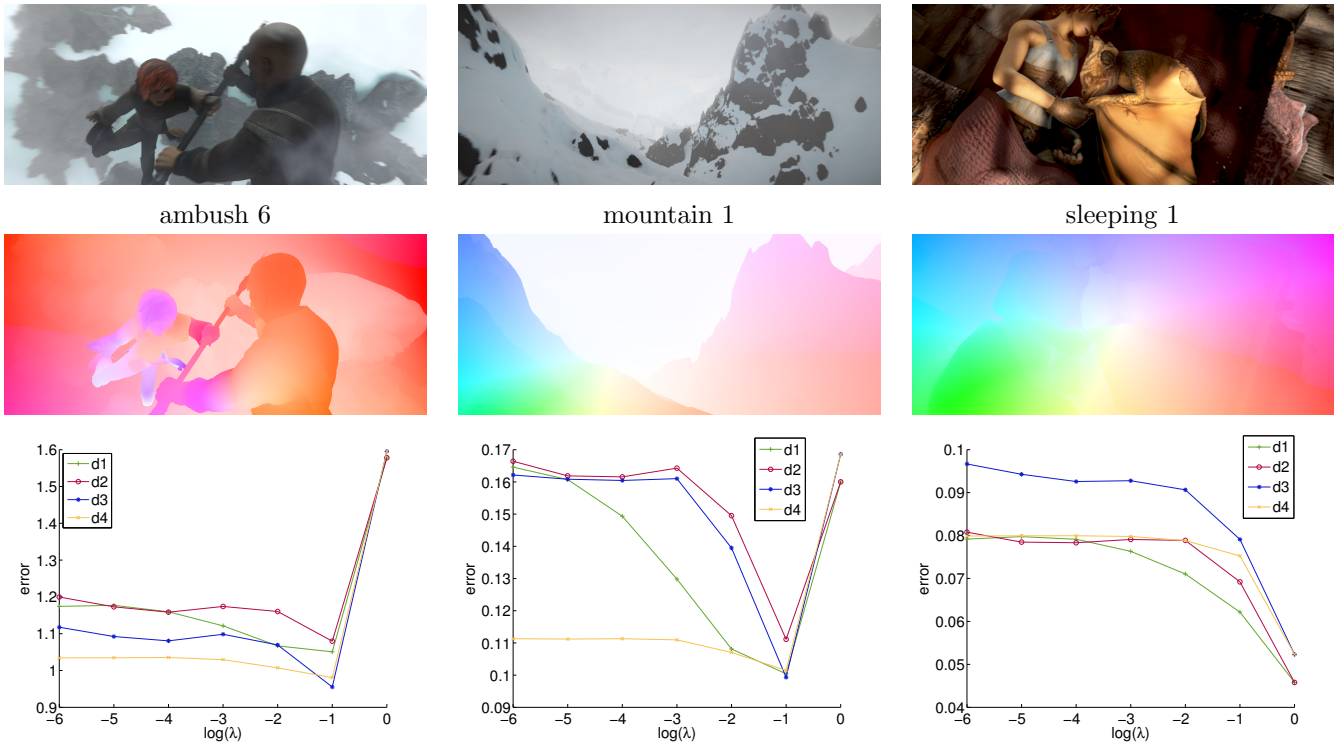

Figure 3: AMLE interpolation error for different values of $\lambda \in \{1, 10^{-1}, 10^{-2}, 10^{-3}, 10^{-4}, 10^{-5}, 10^{-6}\}$ and weights ω_i, $i \in \{1, 2, 3\}$. Three examples are shown using a sparse mask with 1% points. From top to bottom: reference frame, ground truth flow and plots of the interpolation error. Each plot indicates the interpolation error for the values of λ and for those of the weights. The optimal value of λ is different for each example and for each weight.

inpainting results obtained using four different guides: a binary shape guide, where the main object is in white and all the background in black (first column in Figures 4 and 5); a cartoon guide, which has no texture (second column in Figures 4 and 5); a textured guide (third column in Figures 4 and 5); and a realistic guide, this last one may contain texture, blur and shadows among others as it is the case for the Final Sintel video sequences [6] (fourth column in Figures 4 and 5).

One can observe how the results obtained with the shape and cartoon guiding images in Figures 4 and 5 better preserve the motion boundaries than the other two results thanks to the contrasted and sharper object edges of the guiding images. When the guiding image contains textures one can observe how these small details affect the color difference and, therefore, produce wrong motion boundaries aligned with the texture details (e.g. third column of Figure 4). In contrast with the previous behaviour, it may happen that the colors among neighboring objects are very close, which produces small values in the geodesic distance and optical flow leaking is visible (e.g. third and fourth columns of Figure 5, where the color of the building is very close to the color of the hair).

2.4 Neighborhood Type

As mentioned in [17] the solution of the numerical scheme defined in (6) converges to the solution of (1) when the local spatial resolution dx (7) and the local directional resolution $d\theta$ (8) tend to 0. In order to satisfy these conditions two different types of neighborhoods $\mathcal{N}(\mathbf{x})$ were evaluated. Let us consider the square of size $(2\,r+1) \times (2\,r+1)$ centered at $\mathbf{x}$. In the first type of neighborhood $\mathcal{N}_1(\mathbf{x})$ (first row of Figure 6) for each possible direction in the square the point at lowest distance from $\mathbf{x}$ is kept. For the second type $\mathcal{N}_2(\mathbf{x})$ (second row of Figure 6), all the points in the square are kept. These two options have the same local directional resolution $d\theta$ but differ in the local spatial resolution dx. The influence of the discretization of (1) is illustrated with a particular example. Let us consider

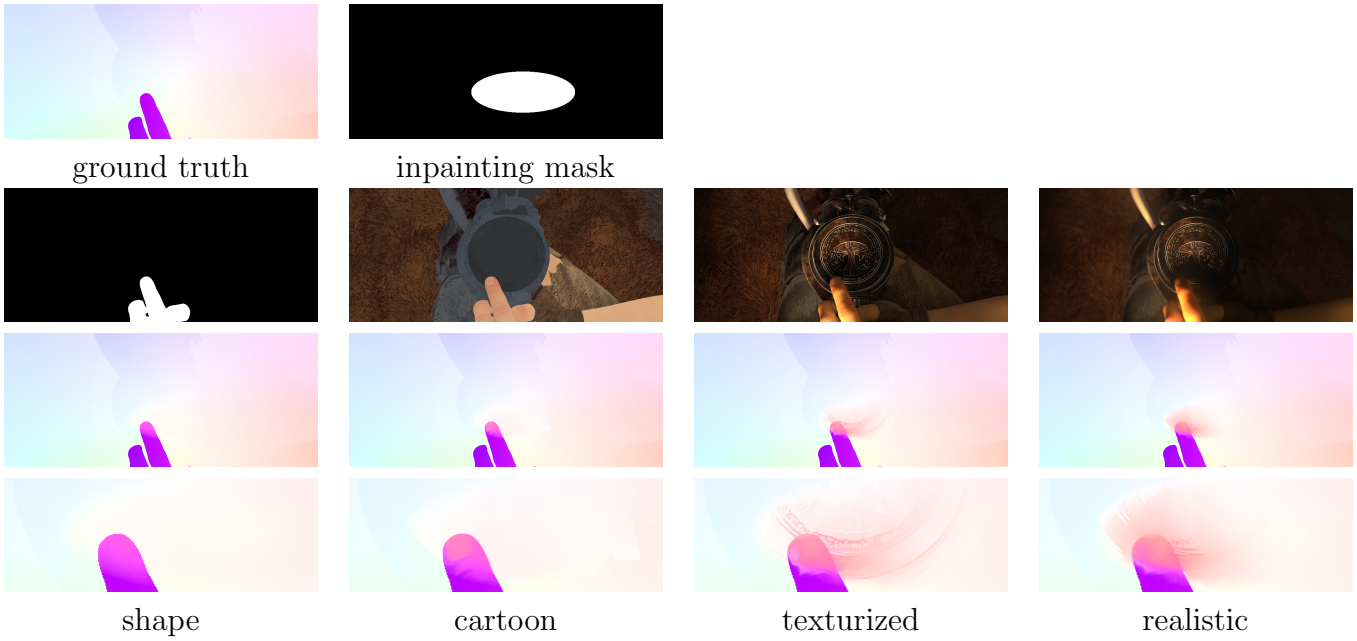

Figure 4: AMLE interpolation. Influence of the metric using different guiding images. First row: the ground truth flow and the inpainting mask where the known optical flow values are in black. Second row: guiding images (from left to right: shape, cartoon, texturized and realistic). Third row: corresponding inpainting results. Fourth row: a zoom-in of the inpainting results.

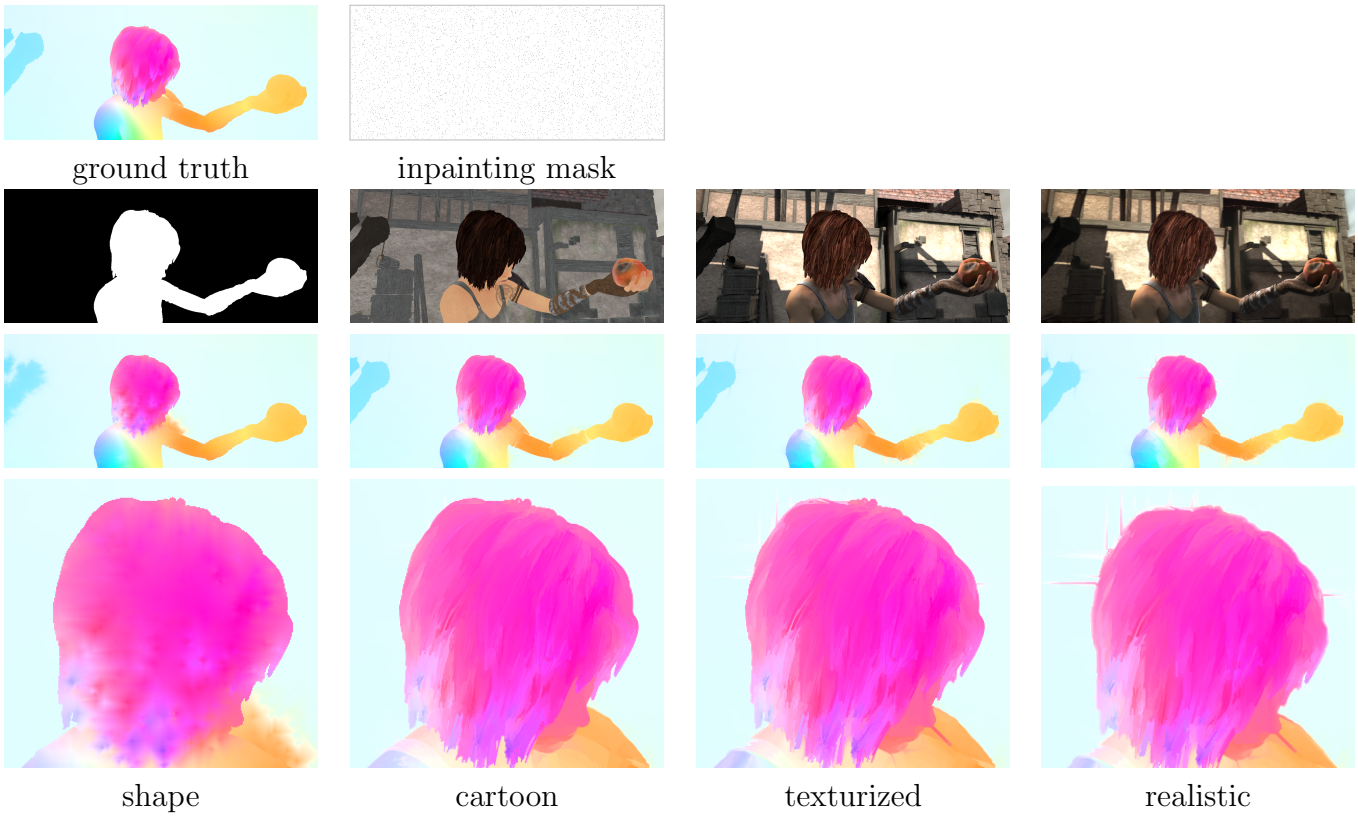

Figure 5: AMLE interpolation. Influence of the metric using different guiding images. First row: the ground truth flow and the inpainting mask containing 1% of known optical flow values in black. Second row: guiding images (from left to right: shape, cartoon, texturized and realistic). Third row: corresponding inpainting results. Fourth row: a zoom-in of the inpainting results.

the ball $B_R((0,0))$ in $\mathbb{R}^2$ of center $(0,0)$ and radius R. Given a function U whose only known values are $U(0,0) = C$, with C a positive constant value, and $U(x,y) = 0$ for $(x,y) \in \partial B_R((0,0))$, the goal is to interpolate U by solving $\Delta_{\infty,g} U = 0$ on $B_R((0,0))$, with g the Euclidean metric, and with those conditions on $(0,0)$ and $\partial B_R((0,0))$. The exact solution to this AMLE problem is the cone of height C and radius R [8]. In Figure 7 several results obtained with the discretization (6) using different types and sizes of neighborhoods are shown. One can observe that in both neighborhood types the approximation to the cone is better recovered when both the spatial and directional resolutions are small which is the case of the third column in Figure 7. The first two columns have a directional resolution which is too large: 45° and 90°. In the fourth and fifth columns, even though the directional resolution is smaller than in the first three cases the spatial resolution gets too large. As both neighborhoods produce similar results and the optical flow interpolation using $\mathcal{N}_1(\mathbf{x})$ is faster (for a fixed r it involves less pixels and, therefore less computations), we choose the first neighborhood configuration with $r = 2$ for all the experiments of the paper, which implies that the cardinality N of $\mathcal{N}_1(\mathbf{x})$ equals 16.

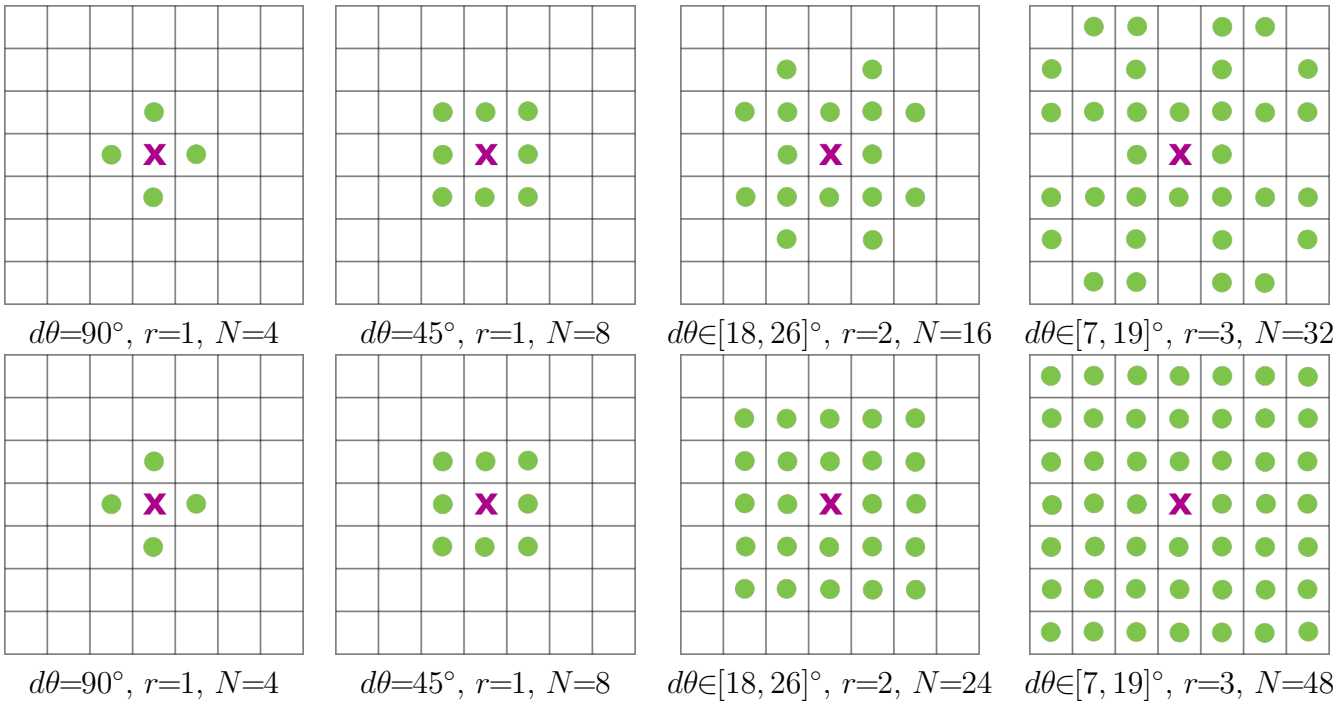

Figure 6: Comparison of the two types of neighborhood considered in the discretization of the geodesic AMLE: $\mathcal{N}_1(\mathbf{x})$ (top) and $\mathcal{N}_2(\mathbf{x})$ (bottom). For each case the square ratio r, the directional resolution $d\theta$ and the cardinality N of $\mathcal{N}(\mathbf{x})$ are shown.

2.5 Multiscale Scheme

The discrete scheme defined in (14) used to solve (1) is embedded in a multiscale approach: the input optical flow and the corresponding video frame are downscaled to a set of scales and the solution is computed at each one using (14). The inputs are downsampled by a factor of two. At the coarsest level, the unknowns are initialized to zero; the other scales are initialized by upsampling the solution of the previous scale. As shown in [8] the AMLE PDE defined in (1) has a unique solution, thus the solution using the single scale or the multiscale approach is the same. This can be observed in the single scale and multiscale results in Figure 9, for the example in Figure 8. The multiscale pyramid simply provides an initialization of the numerical scheme closer to the solution, leading to a global faster convergence. This is illustrated in Figure 10, where one can observe that the multiscale

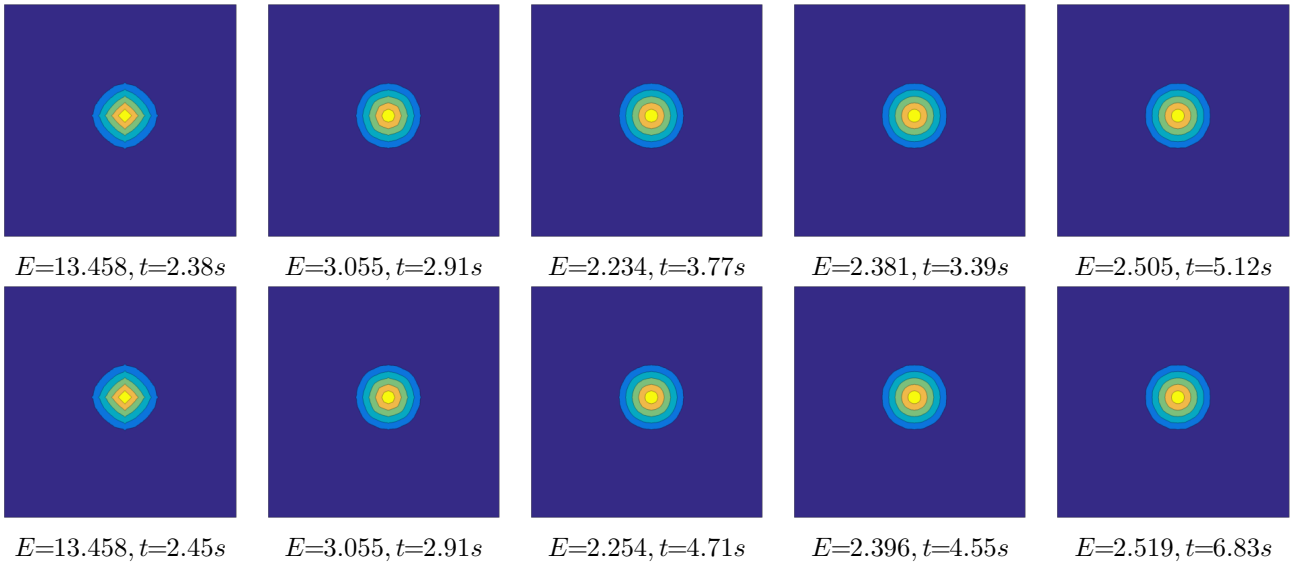

Figure 7: This figure shows different results when interpolating a cone using the AMLE. Top row: from left to right results using $\{4, 8, 16, 32, 64\}$ neighbors, respectively, for neighborhood $\mathcal{N}_1(\mathbf{x})$. Bottom row: from left to right results using $\{4, 8, 24, 48, 80\}$ neighbors for neighborhood $\mathcal{N}_2(\mathbf{x})$. In all cases the interpolation error E and the execution time t in seconds are indicated.

scheme is less time consuming. For this, the runtime and the interpolation error E defined in (15) are plotted. One can observe that, for a given interpolation error, the runtime of the multiscale algorithm is smaller than the one of the single scale case.

Figure 8: Left: ground truth optical flow. Center: inpainting mask where white pixels indicate the inpainting region given by the occlusion areas. Right: guiding image.

Figure 9: AMLE interpolation. From left to right: the inpainting result using one scale and using four scales for the example in Figure 8. The interpolation error converges to the same value for different numbers of iterations and both inpainting results are indistinguishable.

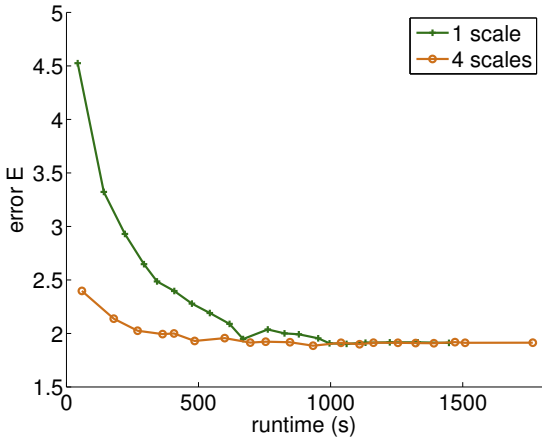

Figure 10: AMLE interpolation. Comparing the execution time of the single and multiscale algorithms.

3 Inpainting using the Laplace-Beltrami Operator

The Laplace-Beltrami (LB) operator can also be used for anisotropic interpolation in Riemannian manifolds. In this section the reconstruction of $v(\mathbf{x},t)$ in $\Omega_0(t)$ is done by solving the Laplace-Beltrami PDE with Dirichlet boundary conditions on $\partial\Omega_0(t)$. Actually, the values on $\Omega \setminus \Omega_0(t)$ are prescribed. The Laplace-Beltrami operator is usually denoted by $\Delta_{2,g}$, or simply by Δ_g, and defined by

$$\Delta_g u(\mathbf{x}) = \text{Trace}\left(G^{-1}(\mathbf{x}) D^2_{\mathcal{M}} u\right) = \text{Trace}\left(G^{-1}(\mathbf{x})\left(\frac{\partial^2 u}{\partial x^i \partial x^j} - \Gamma^k_{ij}\frac{\partial u}{\partial x^k}\right)\right), \tag{16}$$

where $G(\mathbf{x})$ denotes the (symmetric) matrix $(g_{ij}(\mathbf{x}))$. Or, equivalently,

$$\Delta_g u(\mathbf{x}) = \frac{1}{\sqrt{\det(G(\mathbf{x}))}}\partial_i\left(\sqrt{\det(G(\mathbf{x}))}g^{ij}\partial_j u\right). \tag{17}$$

As before, let us forget about the time variable and write Ω_0 and $\partial\Omega_0$ instead of $\Omega_0(t)$ and $\partial\Omega_0(t)$, respectively. Thus, the reconstructed optical flow is given by $u(\mathbf{x},t)$ such that u_1 and u_2 are the solutions of

$$\begin{cases} \Delta_g u_c = 0 & \text{in } \Omega_0 \\ u_c = v_c, & \text{in } \partial\Omega_0. \end{cases} \tag{18}$$

3.1 The Discrete Model

As in Section 2.1, the discrete grid of Ω (with grid size $h = 1$ in x and y direction) is considered as the nodes or vertices of a finite weighted graph $G = (V, \mathcal{E}, \omega)$, where V is the set of vertices, $\mathcal{E} \subset V \times V$ is the set of edges between pairs of vertices, and $\omega : \mathcal{E} \to \mathbb{R}$ denotes the weights on the edges, as defined in Section 2.1. Let us assume that $n = |V|$ and $m = |\mathcal{E}|$ are the cardinals of V and $\mathcal{E}$, respectively. In the following we will assume that n is the number of pixels of $(\Omega_0 \cup \partial\Omega_0) \cap \mathbb{Z} \times \mathbb{Z}$.

Indeed, the discretization of (18) is done using this graph formalism and then the PDE problem translates into a linear problem where the linear matrix has size $n \times n$. To do so, let us use the usual compact notation where the double indexing in each pixel $(i,j) \in (\Omega_0 \cup \partial\Omega_0) \cap \mathbb{Z} \times \mathbb{Z}$ is replaced by a single index and where all unknown gray values are assembled in a one-dimensional vector $\mathbf{u} \in \mathbb{R}^n$. Let us recall that usually in this context, the Laplacian operator Δ (that is, when g denotes the Euclidean metric) can be expressed as a square matrix L, of size n [11, 14], which is defined as

$$L = -B^T B, \tag{19}$$

where B is the incidence matrix of the graph and has m rows and n columns. The incidence matrix B represents the gradient operator and $-B^T$ the divergence operator. In other words, (19) corresponds to the definition of the Laplacian operator $\Delta = div(\nabla)$ in the discrete context.

As mentioned earlier, the motion reconstruction problem is solved using an anisotropic interpolation with the aim of respecting the moving object boundaries. For that, the domain Ω is endowed with a metric $g(t)$ at the frame domain at time t. Then, the Laplace-Beltrami operator Δ_g can be expressed in matrix form as

$$L_g = -B^T W B, \tag{20}$$

where W is an $m \times m$ diagonal matrix containing the weights defined by one of the proposed metrics ((10), (11) or (12)) in the manifold [11, 14]. These weights are defined for every edge $(\mathbf{x}, \mathbf{y}) \in \mathcal{E}$ between two adjacent vertices $\mathbf{x}$ and $\mathbf{y}$ of the graph as

$$\omega(\mathbf{x}, \mathbf{y}) = \frac{1}{d(\mathbf{x}, \mathbf{y})}. \tag{21}$$

Finally, adding the boundary conditions, problem (18) becomes

$$\begin{cases} L_g \mathbf{u} = 0 & \text{in } \Omega_0 \\ \mathbf{u} = \mathbf{v}_c & \text{in } \partial\Omega_0, \end{cases} \tag{22}$$

which can be re-formulated as the linear system

$$(ML_g + I - M)u_c = (I - M)v_c, \tag{23}$$

where M is an $n \times n$ diagonal matrix of ones and zeros equal to the indicator $\mathbb{I}_{\Omega_0}$ of the subset Ω_0, with diagonal values equal to one on the pixels to inpaint, and I is the identity matrix of size $n \times n$. The following result guarantees that the linear system (23) has a unique solution and it is given by

$$u_c = (ML_g + I - M)^{-1}(I - M)v_c, \ c = 1, 2. \tag{24}$$

Theorem *The linear system* (23) *has a unique solution, for each* $c = 1, 2$*, which is given by* (24)*. Moreover, the solution satisfies in* $\Omega_0 \cap \mathbb{Z} \times \mathbb{Z}$ *the following maximum and minimum principle*

$$\min_{\tilde{k} \in \partial\Omega_0 \cap \mathbb{Z} \times \mathbb{Z}} v_c(\tilde{k}) \leq u_c(k) \leq \max_{\tilde{k} \in \partial\Omega_0 \cap \mathbb{Z} \times \mathbb{Z}} v_c(\tilde{k}) \qquad \forall k \in \Omega_0 \cap \mathbb{Z} \times \mathbb{Z}, \tag{25}$$

for $c = 1, 2$.

The proof of this result can be found in [15] (Theorem 1).

3.2 An Approximation of d^g

As in Section 2.2 the same weights are considered, which are defined in (10), (11) and (12). In Figures 11 and 12 one can observe that in the case of the Laplace-Beltrami interpolation the weight ω_3 is again the one that gives better results, both qualitatively and quantitatively. Also, from Figure 13 one can once again observe that there is no unique optimal value of λ but rather an optimal one for each image and weight $\omega_i(\mathbf{x}, \mathbf{y})$.

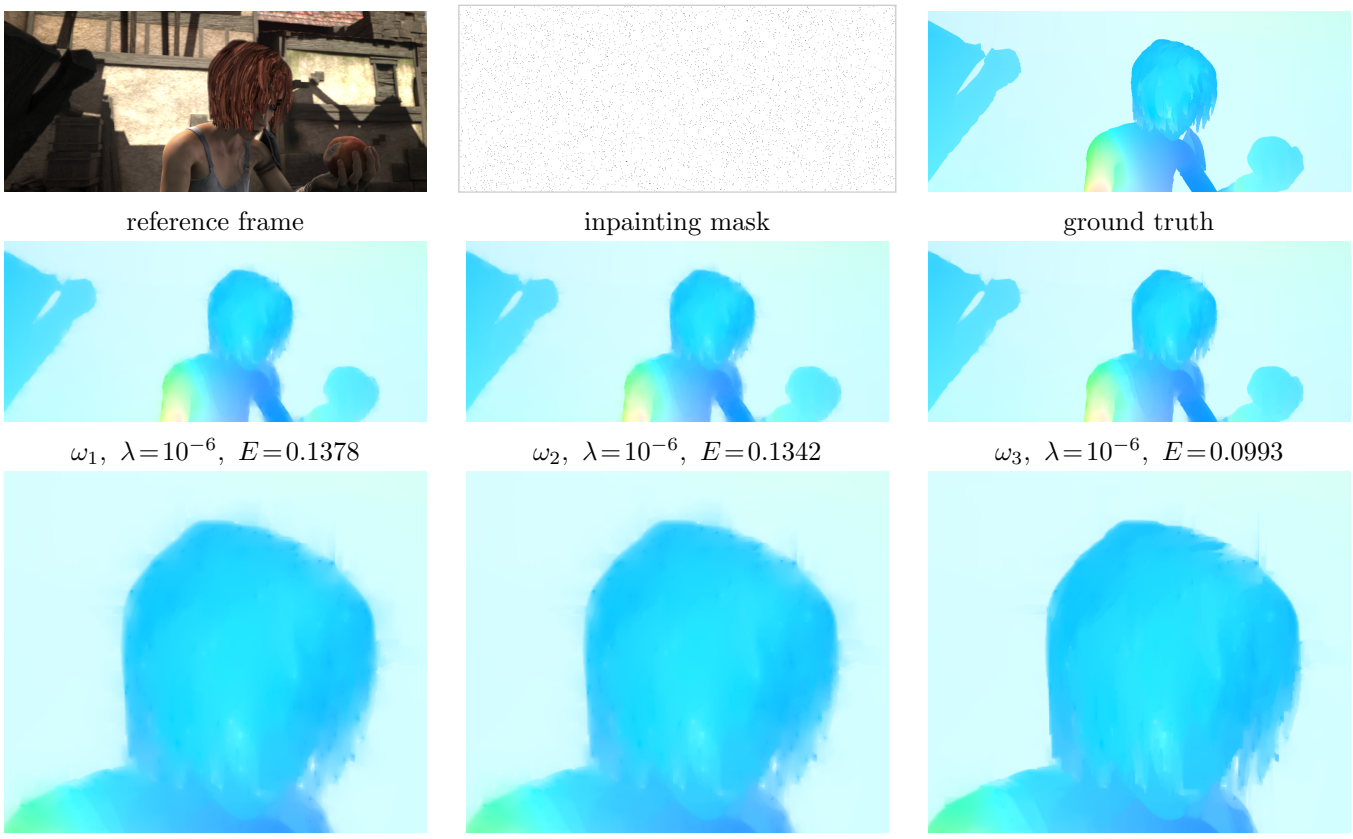

Figure 11: Laplace-Beltrami interpolation. Influence of the metric. First row, from left to right, the guiding image, the sparse inpainting mask with 1% points where white pixels denote unknown optical flow values, and the ground truth flow. Second row shows four inpainting results corresponding, from left to right, to the weights $\omega_1(\mathbf{x},\mathbf{y})$, $\omega_2(\mathbf{x},\mathbf{y})$ and $\omega_3(\mathbf{x},\mathbf{y})$. The interpolation error and the value of λ are indicated for each result. The last row shows a detail of the results of the second row.

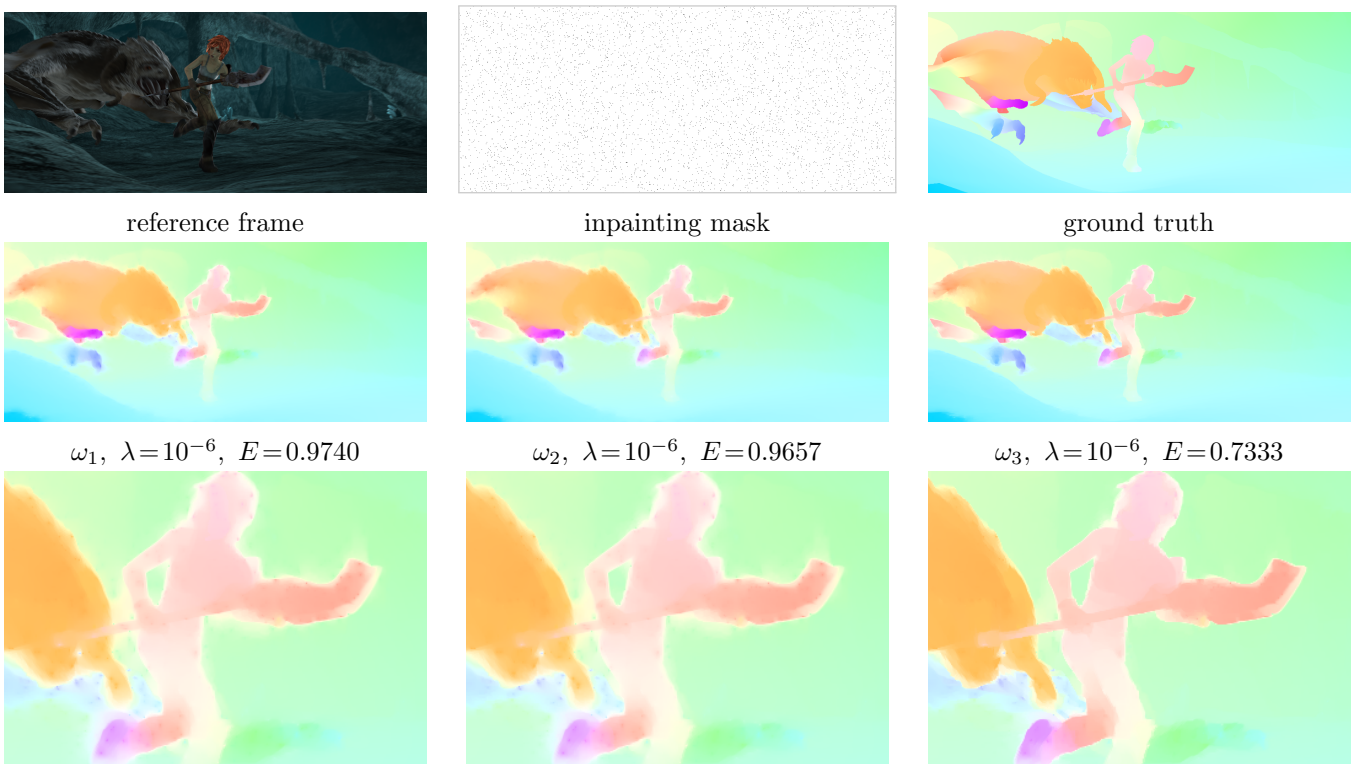

Figure 12: Laplace-Beltrami interpolation. Influence of the metric. First row, from left to right, the guiding image, the sparse inpainting mask with 1% points where white pixels denote unknown optical flow values, and the ground truth flow. Second row shows four inpainting results corresponding, from left to right, to the weights $\omega_1(\mathbf{x},\mathbf{y})$, $\omega_2(\mathbf{x},\mathbf{y})$ and $\omega_3(\mathbf{x},\mathbf{y})$. The interpolation error and the value of λ are indicated for each result. The last row shows a detail of the results of the second row.

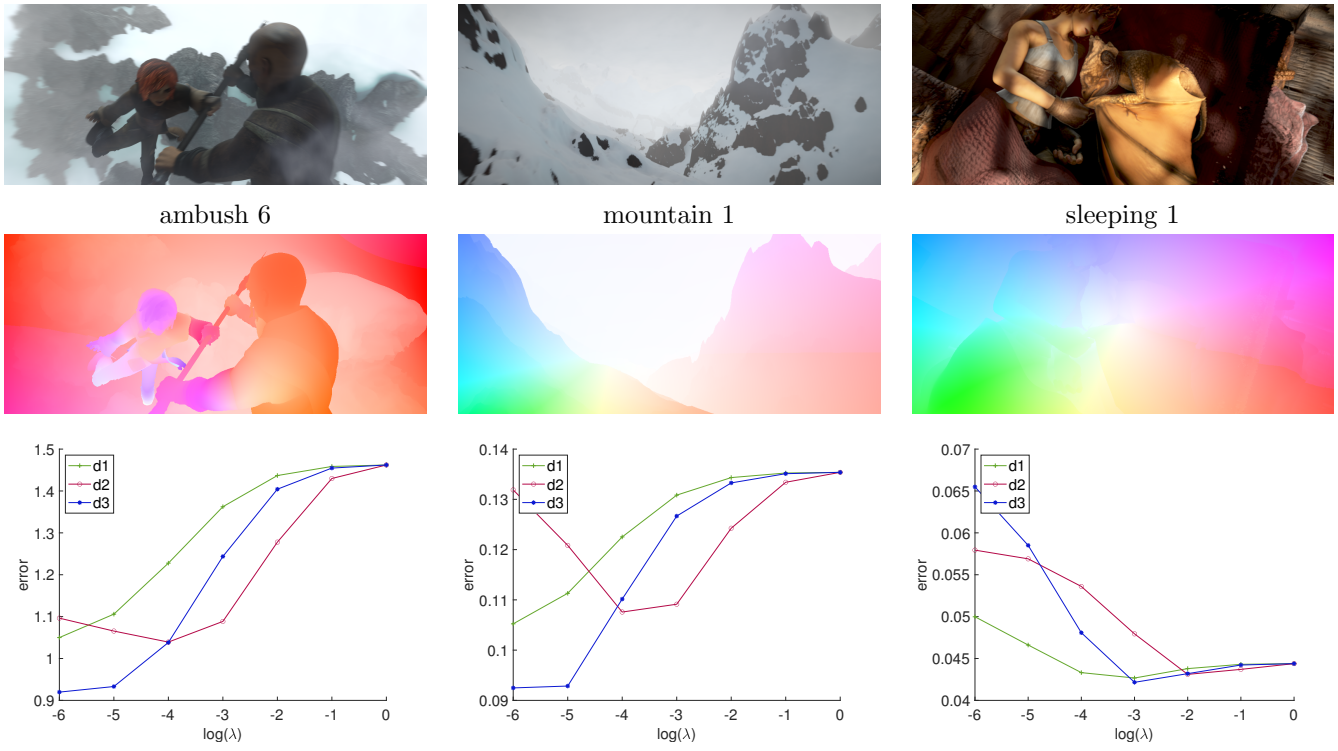

Figure 13: Laplace-Beltrami interpolation error for different values of $\lambda \in \{1, 10^{-1}, 10^{-2}, 10^{-3}, 10^{-4}, 10^{-5}, 10^{-6}\}$ and weights $\omega_i,\ i \in \{1,2,3\}$. Three examples are shown using a sparse mask with 1% points. From top to bottom: reference frame, ground truth flow and plots of the interpolation error. Each plot indicates the interpolation error for the values of λ and for those of the weights. The optimal value of λ is different for each example and for each weight.

4 Explanation of the Parameters

The parameters of both methods are described in this section. The geodesic AMLE interpolation algorithm depends on four parameters: the anisotropic weight (λ), the stopping criterion threshold (ε), the square ratio r defining the size of the local neighborhood and the number of scales S used to build the flow pyramid. The Laplace-Beltrami interpolation algorithm only depends on the anisotropic weight (λ).

- λ is the anisotropic weight used in definitions (10), (11), (12) and (13), which takes values in $[0, 1]$. It determines the degree of anisotropy of the method and it is inversely proportional to the anisotropic diffusion. When $\lambda = 1$ the Euclidean metric is used and the AMLE and the Laplace operators are recognized for the cases ω_1 and ω_2. The experiments show that a reasonable default value is 0.001.

- ε is the stopping criterion threshold used in the numerical scheme which is a trade-off between precision and running time. Small values will yield more precise results at the expense of higher running times. A reasonable default value is $\varepsilon = 0.0001$.

- r is the square ratio which defines the size of the local square neighborhood $\mathcal{N}(\mathbf{x})$ of a node $\mathbf{x}$ on the graph. It defines a trade-off between the local spatial and directional resolutions defined in (7) and (8) respectively. When $r = 1$ the local spatial resolution is small ($dx = 1$) at the expense of a high directional resolution ($d\theta = \pi/4$). Increasing r will decrease $d\theta$ but increase dx. A good default value is $r = 2$.

- S is the number of scales used to build the flow pyramid. It allows to accelerate the convergence of the method. Different values of S yield the same inpainting result. The default value is $S = 4$.

In addition, there are two hyperparameters g and n_t enabling the choice of different weights and the choice of different neighborhoods respectively.

- g indicates which weight will be used for the anisotropic interpolations. In the case of the AMLE in a manifold the possible values of g are $\{1, 2, 3, 4\}$, indicating that the metric used is (10), (11), (12) or (13), respectively. In the case of the Laplace-Beltrami the possible values of g are $\{1, 2, 3\}$, indicating that the metric used is (10), (11) or (12), respectively. The default value is set to $g = 3$ for both cases.

- n_t indicates the type of local neighborhood $\mathcal{N}(\mathbf{x})$ used in the discrete scheme. There are two options: $n_t = 1$ for the neighborhood type $\mathcal{N}_1$ and $n_t = 2$ for $\mathcal{N}_2$, both defined in Section 2.4. The value of n_t conditions the value of the size of the neighborhood N. For $n_t = 1$ and $r \in \{1, 2, 3, 4, 5\}$ the cardinality of $\mathcal{N}_1$ is $N \in \{4, 8, 16, 24, 32\}$, respectively. For $n_t = 2$ and $r \in \{1, 2, 3, 4, 5\}$ the cardinality of $\mathcal{N}_2$ is $N = \{4, 8, 24, 48, 80\}$. The two options yield similar results. However, for $r > 1$ and $n_t = 1$ the execution time is slightly smaller than for $n_t = 2$ since it involves less pixels. The default value is set to $n_t = 1$.

5 The Algorithms

5.1 The AMLE in a Manifold Interpolation

Algorithm 1 provides a general description of the optical flow inpainting approach using the geodesic AMLE in two main steps: computation of the weight map $\tilde{W}$ and application of the iterative discrete

scheme to solve the PDE defined in (1). A detailed pseudocode specifies each of these steps. The method takes as input an incomplete optical flow $\mathbf{v}$, an inpainting mask κ and a guiding image I, which defines the weights of the geodesic interpolation. Three parameters and two hyperparameters are required: the anisotropic weight λ, a stopping criterion threshold ε, the square ratio r, the neighborhood type n_t and the metric g. The output of the method is a complete optical flow $\mathbf{u}$.

The main algorithm described in Algorithm 1 starts by defining the inpainting domain Ω_0 based on the mask κ where positive values indicate the region to inpaint. Then, the relative neighboring pixels positions (dx, dy) for any node (i, j) in the graph are computed and stored in $\mathcal{N}$ where a neighbor of (i, j) is defined as $(i+dx, j+dy)$. The positions in $\mathcal{N}$ depend on the type of neighborhood n_t and the neighborhood ratio r. After that, the weight map $\tilde{W}$ is computed containing for every pixel $(i, j) \in \Omega_0$ the weights of the edges $((i, j), (i + dx, j + dy))$ for all $(dx, dy) \in \mathcal{N}$. Then, the iterative discrete scheme defined in (21) is applied separately to each component $v_c,\ c = 1, 2$, of the flow $\mathbf{v}$ returning the component $u_c,\ c = 1, 2$, of the flow $\mathbf{u}$. An image *init* indicates how to initialize the values of u_1 and u_2 in the inpainting domain Ω_0 and it is given as input to amle_interpolation$(v_c, \tilde{W}, init, \Omega_0, \lambda, \varepsilon)$.

Algorithm 1: Optical flow inpainting (single scale)

input : An incomplete optical flow $\mathbf{v}$, an inpainting mask κ, a guiding image I, an anisotropic weight λ, a stopping criterion threshold ε, a weight choice g, a neighborhood type n_t and a neighborhood ratio r

output: A completed optical flow $\mathbf{u}$

$\Omega_0 \leftarrow \{(i, j) \in \Omega : \kappa[i, j] \geq 0\}$ *inpainting domain.*

$\mathcal{N} \leftarrow$ get_neighboring_indexes(n_t, r)

$\tilde{W} \leftarrow$ compute_weight_map$(I, \lambda, g, \Omega_0, \mathcal{N})$ *Algorithm 2*

for $(i, j) \in \Omega$ **do**

  $init[i, j] \leftarrow 0$ *initialization image of 0's*

for $c \in \{1, 2\}$ **do**

  $u_c \leftarrow$ amle_extension$(v_c, \tilde{W}, init, \Omega_0, \lambda, \varepsilon)$ *Algorithm 3*

$u \leftarrow [u_1; u_2]$

return u

The computation of the weight map is described in Algorithm 2. It takes as input a guiding image I, the anisotropic weight λ, the variable g, the inpainting region Ω_0 and the relative positions of the neighboring pixels in $\mathcal{N}$. The variable g takes values in $\{1, 2, 3, 4\}$, indicating which type of weights will be used. Overall, for each pixel (i, j) in the inpainting domain Ω_0, the weights to its N neighboring pixels $(x, y) = (i + dx, j + dy)$ are computed and stored in $\tilde{W}[i, j, p],\ p \in \{0, \ldots, N - 1\}$. The matrix of weights $\tilde{W}$ is of size $|\Omega_0| \times N$.

Algorithm 3 is the core of the method and details the iterative scheme that solves (1). The input is one component v of an optical flow and the output is the respective interpolated component u. The component u is initialized with the values in *init* for all $(i, j) \in \Omega_0$. The iterative scheme (14) is applied until the stopping criterion is reached consisting of a maximum number of iterations $NITER$ and a threshold ε on the error e_k between two consecutive iterations. The value of $NITER$ is set to 5000. Algorithm 4 implements one iteration of the discrete scheme where the values of one component of the optical flow are updated only for the pixels (i, j) in Ω_0. For each node $(i, j) \in \Omega_0$ the positive and negative eikonals are computed as defined in (4). That is, the node (x_p, y_p) and (x_n, y_n) maximize and minimize the derivatives in (i, j) respectively. These two nodes define the pixels used to interpolate the value of u in pixel (i, j) at iteration k following (14).

The iterative discrete scheme, as commented in Section 2.5 converges very slowly to the solution of (1), in particular when the inpainting domain is large. To accelerate the convergence a multiscale

Algorithm 2: compute_weight_map$(I, \lambda, g, \Omega_0, \mathcal{N})$

input : A guiding image I, an anisotropic weight λ, a weight choice ω, an inpainting domain Ω_0 and neighboring indexes $\mathcal{N}$
output: An image of weights $\tilde{W}$

$C \leftarrow \dim(I)$ *number of channels of the guiding image I.*
foreach $(i, j) \in \Omega_0$ **do**
  foreach $(dx, dy) \in \mathcal{N}$ **do**
    $p \leftarrow 0$ *initialize position in neighboring indexes $\mathcal{N}$.*
    $D[i, j, p] \leftarrow 0$ *initialize the distance image.*
    $x \leftarrow i + dx$ *x component of a neighboring pixel.*
    $y \leftarrow j + dy$ *y component of a neighboring pixel.*
    switch ω **do**
      case *1*
        foreach $c \in C$ **do**
          $D[i, j, p] \leftarrow D[i, j, p] + (I[x, y, c] - I[i, j, c])^2$
        $D[i, j, p] \leftarrow D[i, j, p]/C$
        $D[i, j, p] \leftarrow \sqrt{(1-\lambda)D[i, j, p] + \lambda((x-i)^2 + (y-j)^2)}$ *definition (10)*
        $\tilde{W}[i, j, p] \leftarrow 1/D[i, j, p]$
      case *2*
        foreach $c \in C$ **do**
          $D[i, j, p] \leftarrow D[i, j, p] + (I[x, y, c] - I[i, j, c])^2$
        $D[i, j, p] \leftarrow D[i, j, p]/C$
        $D[i, j, p] \leftarrow (1-\lambda)\sqrt{D[i, j, p]} + \lambda\sqrt{(x-i)^2 + (y-j)^2}$ *definition (11)*
        $\tilde{W}[i, j, p] \leftarrow 1/D[i, j, p]$
      case *3*
        foreach $c \in C$ **do**
          $D[i, j, p] \leftarrow D[i, j, p] + (I[x, y, c] - I[i, j, c])^2$
        $D[i, j, p] \leftarrow D[i, j, p]/C$
        $D[i, j, p] \leftarrow (1-\lambda)D[i, j, p] + \lambda((x-i)^2 + (y-j)^2)$ *definition (12)*
        $\tilde{W}[i, j, p] \leftarrow 1/D[i, j, p]$
      case *4*
        foreach $c \in C$ **do**
          for $(m, n) \in \mathcal{P}_{s \times s}$ **do**
            $D[i, j, p] \leftarrow D[i, j, p] + (I[x+m, y+n, c] - I[i+m, j+n, c])^2$
        $D[i, j, p] \leftarrow D[i, j, p]/C/s^2$
        $D[i, j, p] \leftarrow (1-\lambda)D[i, j, p] + \lambda((x-i)^2 + (y-j)^2)$ *definition (13)*
        $\tilde{W}[i, j, p] \leftarrow 1/D[i, j, p]$
    $p \leftarrow p + 1$ *update position.*
return $\tilde{W}$

Algorithm 3: amle_extension(v, $\tilde{W}$,init, $\Omega_0, \lambda, \varepsilon, \mathcal{N}$)

input : An incomplete optical flow component v, a weight map $\tilde{W}$, an initialization image *init*, an inpainting domain Ω_0, an anisotropic weight λ, a stopping criterion threshold ε, neighboring indexes $\mathcal{N}$
output: A completed optical flow component u

```
for i ∈ Ω do
    if i ∈ Ω_0 then
        u[i] ← init[i]
    else
        u[i] ← v[i]
u_0 ← u
e_k ← 1000                                              initialize e_k
while k < NITER and e_k > ε do
    u ← amle_iteration(u, Ω_0, W̃, N)                    Algorithm 4.
    e_k ← 1/|Ω_0| Σ_{(i,j)∈Ω_0} |u[i,j] − u_0[i,j]|
    u_0 ← u
return u
```

Algorithm 4: amle_iteration($v, \Omega_0, \tilde{W}, \mathcal{N}$)

input : An optical flow component v, an inpainting domain Ω_0, a weight map W, neighboring indexes $\mathcal{N}$
output: An optical flow component u

```
foreach (i,j) ∈ Ω_0 do
    e_p ← −∞                                            initialize positive eikonal.
    e_n ← +∞                                            initialize negative eikonal.
    p ← 0                                               initialize counter.
    foreach (dx,dy) ∈ N do
        x ← i + dx
        y ← j + dy
        e ← (u[x,y] − u[i,j]) W̃[i,j,p]                  relaxed gradient.
        if e > e_p then
            e_p ← e                                     update positive eikonal.
            w_p ← W̃[i,j,p]                              update positive weight.
            [x_p, y_p] ← [x,y]                          update position.
        if e < e_n then
            e_n ← e                                     update negative eikonal.
            w_n ← W̃[i,j,p]                              update negative weight.
            [x_n, y_n] ← [x,y]                          update position.
        p ← p + 1                                       update counter.
    u[i,j] ← (v[x_n,y_n] w_n + v[x_p,y_p] w_p)/(w_n + w_p)   update new value.
return u
```

approach is considered and described in Algorithm 5. This is done in a recursive way calling at each scale Algorithm 5. At the coarsest scale, $s = 1$, the inpainting domain is initialized with zeros and a first optical flow $u^s, (s = 1)$ is interpolated following Algorithm 3. The output u^s of Algorithm 3 is then used as the initialization of the following scale $s+1$ obtained by upsampling u^s by a factor two. Algorithm 3 is then once again applied obtaining $u^s, (s = 2)$ that will be used as the initialization of the following scale. This procedure is repeated until the finest scale $s = S$ is reached, where S is the total number of scales. The inputs are downsampled by a factor of 2 using 2×2 block averages and bilinear interpolation is used for upsampling.

Algorithm 5: amle_recursive$(v, I, \Omega_0, s, \lambda, \varepsilon, \omega, \mathcal{N})$

input : An incomplete optical flow v, a guiding image I, an inpainting domain Ω_0, number of scales s, an anisotropic weight λ, a stopping criterion threshold ε, a weight choice ω and a the neighboring indexes $\mathcal{N}$

output: A completed optical flow u

$\tilde{W} \leftarrow$ compute_weight_map$(I, \lambda, \omega, \Omega_0, \mathcal{N})$ *Algorithm 2*

if $s > 1$ **then**

  $v_s \leftarrow$ zoom_out_2(v) *zoom out of a factor 2*

  $I_s \leftarrow$ zoom_out_2(I) *zoom out of a factor 2*

  $u_s \leftarrow$ amle_recursive$(v_s, I_s, \Omega_0, \lambda, \varepsilon, \omega, s-1)$ *Algorithm 5*

  $init \leftarrow$ zoom_in_2(u_s) *update the initialization image*

else

  for $(i, j) \in \Omega$ **do**

    $init[i, j] \leftarrow 0$ *initialize the coarsest scale wits 0s*

$u \leftarrow$ amle_extension$(v, \tilde{W}, init, \Omega_0, \lambda, \varepsilon, \mathcal{N})$ *Algorithm 3*

return u

5.2 The Laplace-Beltrami Interpolation

Algorithm 6 provides a description of the optical flow inpainting approach using the Laplace-Beltrami interpolation in three main steps: computation of the incidence matrix, computation of the weight map and solving the linear system defined in (23). A detailed pseudo-code specifies each of these steps. The method takes as input an incomplete optical flow $\mathbf{v}$, a color guiding image I and an inpainting mask κ. One parameter and one hyperparameter are required, the anisotropic weight λ and the metric g. The output of this method is a completed optical flow $\mathbf{u}$.

The first step of Algorithm 6 consists in computing the sparse incidence matrix B which represents the gradient operator. For an image of size $h \times w = m$, the matrix B is rectangular of size $m \times n$, where m is the number of edges in the grid graph of the image. In the case of 4-connectivity, we have $m = (w-1)h + (h-1)h$. Therefore, $\nabla z = Bz$ yields the gradient of a grayscale image z. The incidence matrix is then used to compute the three different weight maps W depending on the value of the variable g. W is a vector of size $1 \times m$, where m is the number of edges of the graph. W is reshaped to a diagonal matrix of size $m \times m$. The weighted Laplacian L_w is computed using the incidence matrix B and the weight map matrix W. The last step consists in building and solving the linear system $Ax_c = b_c$, where x_c is the solution in vector shape and is reshaped into a $h \times w$ matrix.

The computation of the incidence matrix is described in Algorithm 7 for a 4-connectivity graph. Three Matlab functions are used: sparse(i,j,s,m,n), speye(m,n) and kron(x,y). The first one builds a sparse matrix of size $m \times n$ such that the value in pixel $(i[k], j[k])$ is equal to $s[k]$. Here i, j and

s are vectors of the same length. The second function builds a sparse identity matrix of size $m \times n$. The third and last function is the Kronecker tensor product of x and y.

Algorithm 6: laplace_beltrami_interpolation$(v, g, \kappa, \lambda, \omega)$

input : An incomplete optical flow v, a color image g, an inpainting mask κ, an anisotropic weight λ, a weight ω
output: A completed optical flow u

$B \leftarrow$ incidence_matrix_4N(h,w) — *Algorithm 7*
$\|\nabla g\|^2 \leftarrow (Bg_R \odot Bg_R + Bg_G \odot Bg_G + Bg_B \odot Bg_B)/3$ — *element wise multiplication*
$\|\nabla X\|^2 \leftarrow x \odot x + y \odot y$
switch ω **do**
  case *1*
    $W \leftarrow \left[\sqrt{(1-\lambda)\|\nabla g\|^2 + \lambda\|\nabla X\|^2}\right]^{-1}$
  case *2*
    $W \leftarrow \left[(1-\lambda)\|\nabla g\| + \lambda\|\nabla X\|\right]^{-1}$
  case *3*
    $W \leftarrow \left[(1-\lambda)\|\nabla g\|^2 + \lambda\|\nabla X\|^2\right]^{-1}$
$W \leftarrow diag(W)$ — *diagonal matrix with W on the principal diagonal*
$L_w \leftarrow -B^t(WB)$ — *weighted Laplacian*
$\kappa_s \leftarrow spdiags(\kappa)$ — *sparse diagonal matrix with κ on the principal diagonal*
$A \leftarrow \mathcal{I}_{wh} - \kappa_s + \kappa_s L_w$ — *$\mathcal{I}_{wh}$ sparse identity matrix of size $wh \times wh$*
for $c \in \{1, 2\}$ **do**
  $b_c \leftarrow (\mathcal{I}_{wh} - \kappa_s)v_c$
  $x_c \leftarrow A^{-1}b_c$
  $u_c \leftarrow reshape(x_c, h, w)$
$u \leftarrow [u_1, u_2]$
return u

Algorithm 7: incidence_matrix_4N(h, w)

input : A number of rows h and a number of columns w
output: A matrix of incidence B

$x \leftarrow sparse(1 : h-1, 2 : h, 1, h-1, h) - speye(h-1, h)$ — *path of length h*
$y \leftarrow sparse(1 : w-1, 2 : w, 1, w-1, w) - speye(w-1, w)$ — *path of length w*
$B \leftarrow [kron(speye(w), x); kron(y, speye(h))]$ — *Kronecker union*
return B

6 Experimental Results and Discussion

In order to compare the performance of the AMLE and the Laplace-Beltrami anisotropic interpolations in Riemannian manifolds, this section illustrates their behaviour with some experiments. For each experiment, the optimal parameters providing the lower EPE error are used. Notice that in some cases, the optimal metrics (i.e. the weights) used in both approaches are not the same. To generate the results of all the experiments shown in this section Algorithms 5 and 6 are used. Two different applications are considered, namely, optical flow densification (Figure 15) and optical flow inpainting, both in occlusion areas (Figure 17) and in large holes (Figure 19). The results will be presented as follows: a reference frame (guiding image), an inpainting mask, the ground truth flow, the output flows and the error images. The error image is computed as the L_2 norm of the difference between the ground truth flow $\bar{\mathbf{v}}$ and the inpainted flow $\mathbf{u}$.

Flow densification A sparse flow is provided containing 1% isolated ground truth flow values (the sparse mask is presented in Figure 14). The geodesic AMLE and the Laplace-Beltrami interpolations are applied to complete the missing data of this flow. Overall, the visual quality of the results of these interpolations are comparable as can be seen in Figure 15. Laplace-Beltrami's result yields a lower EPE since it better recovers smooth regions. The Laplace-Beltrami interpolation at one pixel is a weighted average of its 4-connected neighbors while the geodesic AMLE is the weighted average of two neighboring nodes verifying (4). Moreover, in both cases the diffusion across edges can be visible. This is due to the content of the guiding image where the foreground object and the background have similar intensities. When isolated ground truth points are provided the geodesic AMLE interpolation performs better in the neighborhood of these points. Indeed, as it is well known, the AMLE operator appropriately deals with the Dirichlet data given on isolated points producing a Lipschitz interpolator from these isolated data. This is not the case of the Laplace-Beltrami operator. For example, in the first row of Figure 15, in the region of the body and shirt, one can notice the black isolated points surrounded by a light neighborhood in the Laplace-Beltrami result and not in the geodesic AMLE. To better observe this behaviour the reader is encouraged to zoom-in the results. Notice that, in the error image, dark parts correspond to a low error and bright parts to a high error.

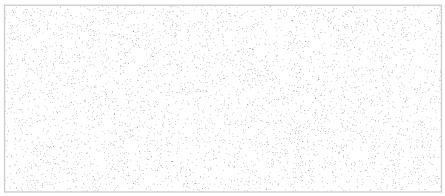

Figure 14: Sparse mask with 1% pixels known (in black).

Flow inpainting: occlusions and large holes Figure 17 shows some results on recovering the missing flow of occluded areas. For each example, a mask containing the occluded regions is provided in Figure 16 where the missing pixels are painted in white. Both the geodesic AMLE and the Laplace-Beltrami interpolations are used to recover the missing flow. The AMLE on a manifold yields slightly better results than the Laplace-Beltrami, both quantitative and qualitatively; e.g. larger error in the right boundary of the foreground person in the first and second examples of Figure 17. Figure 19 shows some results on recovering the missing flow of large holes. The inpainting masks are provided in Figure 18. For large inpainting regions the Laplace-Beltrami interpolation tends to oversmooth the result loosing some of the edges as can be seen in the first and third examples of Figure 19 where the boundary of the head is smoothed out and the wing of the dragon is not fully reconstructed. On

the other hand, the geodesic AMLE produces more diffusion across edges as in the first example in parts where the background and foreground colors are very similar.

An additional observation is related to the fact that the use of the guiding image is not always the best solution. This can be observed in the examples in Figure 20. For instance, when the flow to recover contains gradual discontinuities the results of both the Laplace-Beltrami and geodesic AMLE interpolations are better when no guiding image is used ($\lambda = 1$) as shown in the third and fourth examples of Figure 20. Moreover, when the flow to recover contains sharp edges, as in the two first examples of Figure 20, a guiding image is necessary to control the diffusion across edges.

As can be seen in this section, the qualitative and quantitative results yielded by both methods are very similar. On top of that, the implementation of the Laplace-Beltrami interpolation is reduced to solve a sparse linear system which makes it much faster in terms of execution time regarding the geodesic AMLE which is solved with an iterative discrete scheme. This is why inpainting the missing flows, regardless of the application, with the Laplace-Beltrami operator is a good solution in terms of results and execution time.

7 Conclusions

An implementation of two optical flow inpainting techniques has been described. The first one is based on the AMLE interpolator on a Riemannian manifold and the second one is based on the Laplace-Beltrami operator. Both approaches are based on PDEs that are solved on a 2D Riemannian manifold endowed with an appropriate metric defined by the image frame, which acts as a guiding image for the resulting anisotropic diffusion. The behaviour of these methods were illustrated separately with different examples and different metrics and parameters, and afterwards compared together. In general, both methods perform similarly; the AMLE being more sensitive to color changes and thus producing sharper edges in some situations while causing some leaking when the image colors across a motion boundary are barely distinguishable. From this analysis, one can conclude that the key point relies on the use of non-Euclidean metrics for interpolation rather than the choice of a particular PDE. The use of a good interpolation guide is crucial to define the weights of these anisotropic diffusions.

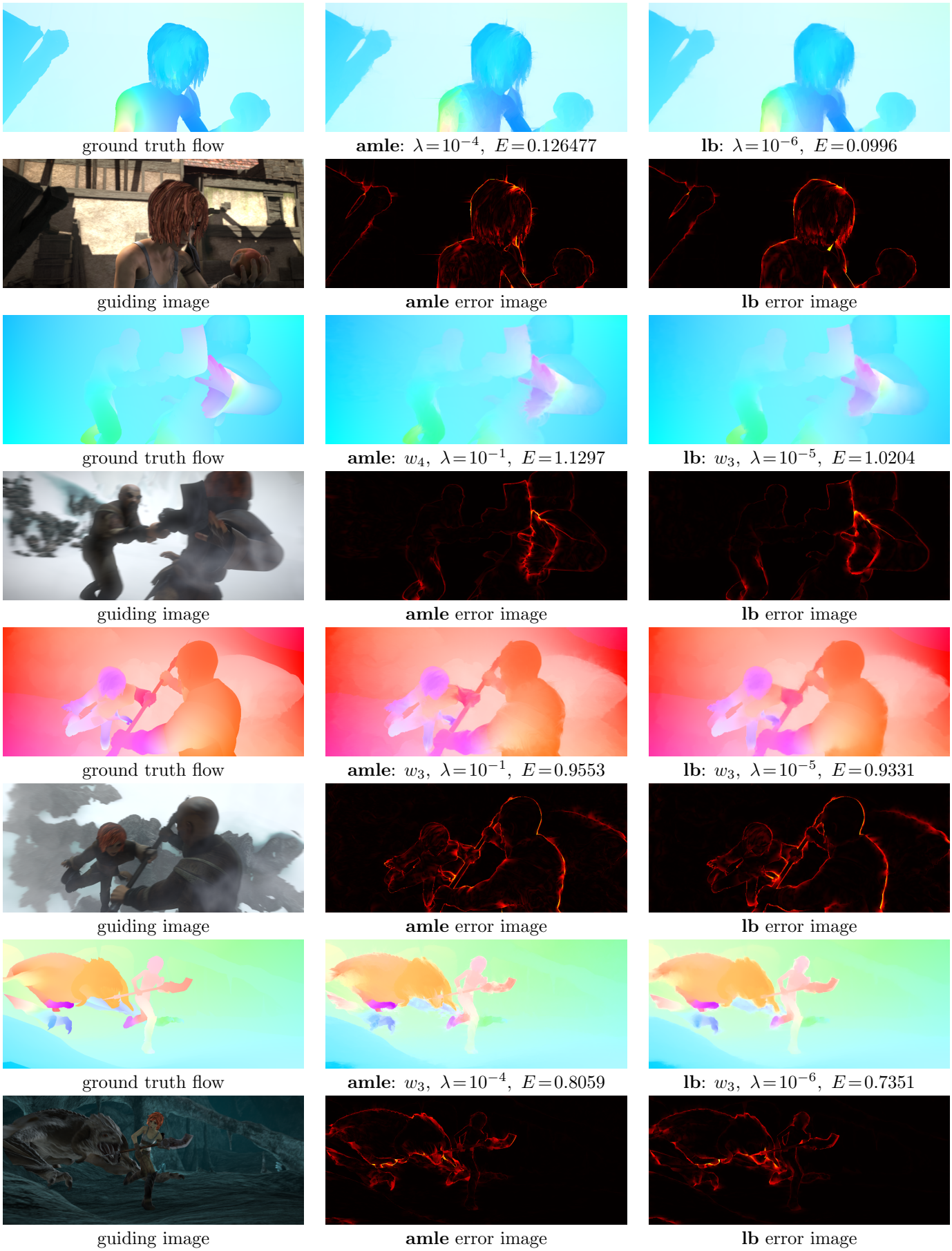

Figure 15: Sparse flow inpainting using the mask in Figure 14. First, third and fifth rows, from left to right: ground truth flow, geodesic AMLE and LB inpainting results. Second, fourth and sixth rows, from left to right: guiding image, geodesic AMLE and LB error images.

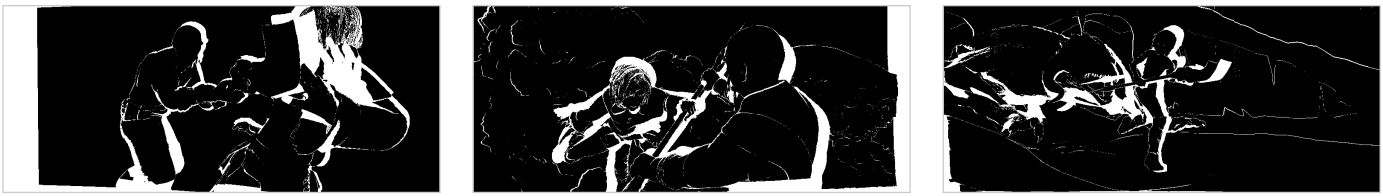

Figure 16: Occlusions masks.

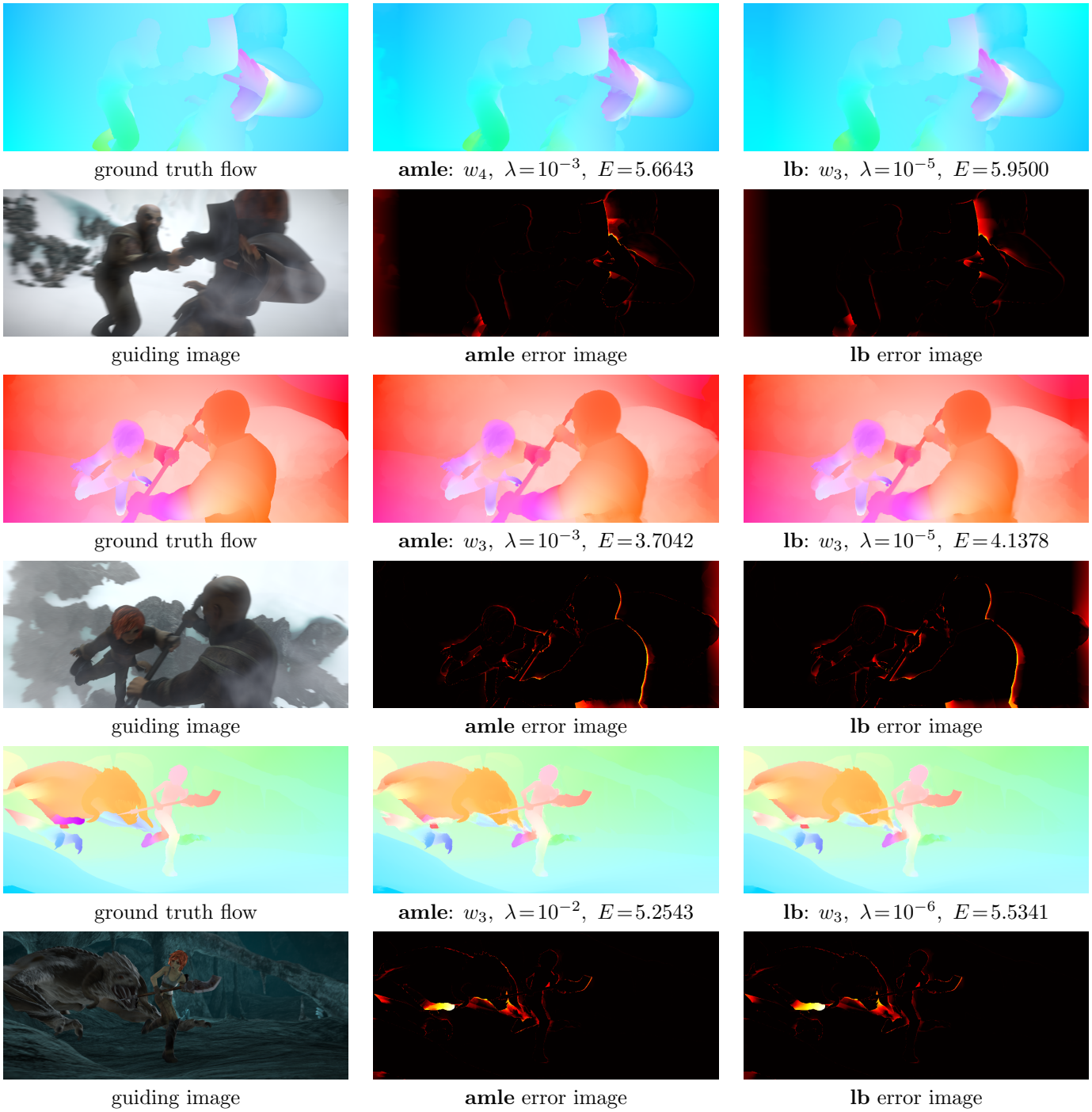

Figure 17: Inpainting of occluded flow using the masks in Figure 16. First, third and fifth row, from left to right: ground truth flow, geodesic AMLE inpainting result, LB inpainting result. Second, fourth and sixth row, from left to right: guiding image, geodesic AMLE and LB error images.

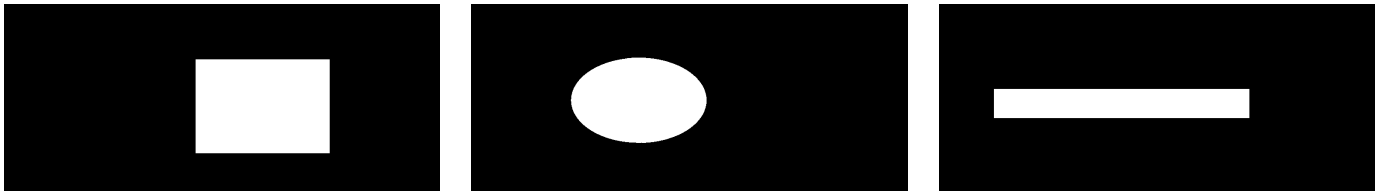

Figure 18: Hole masks.

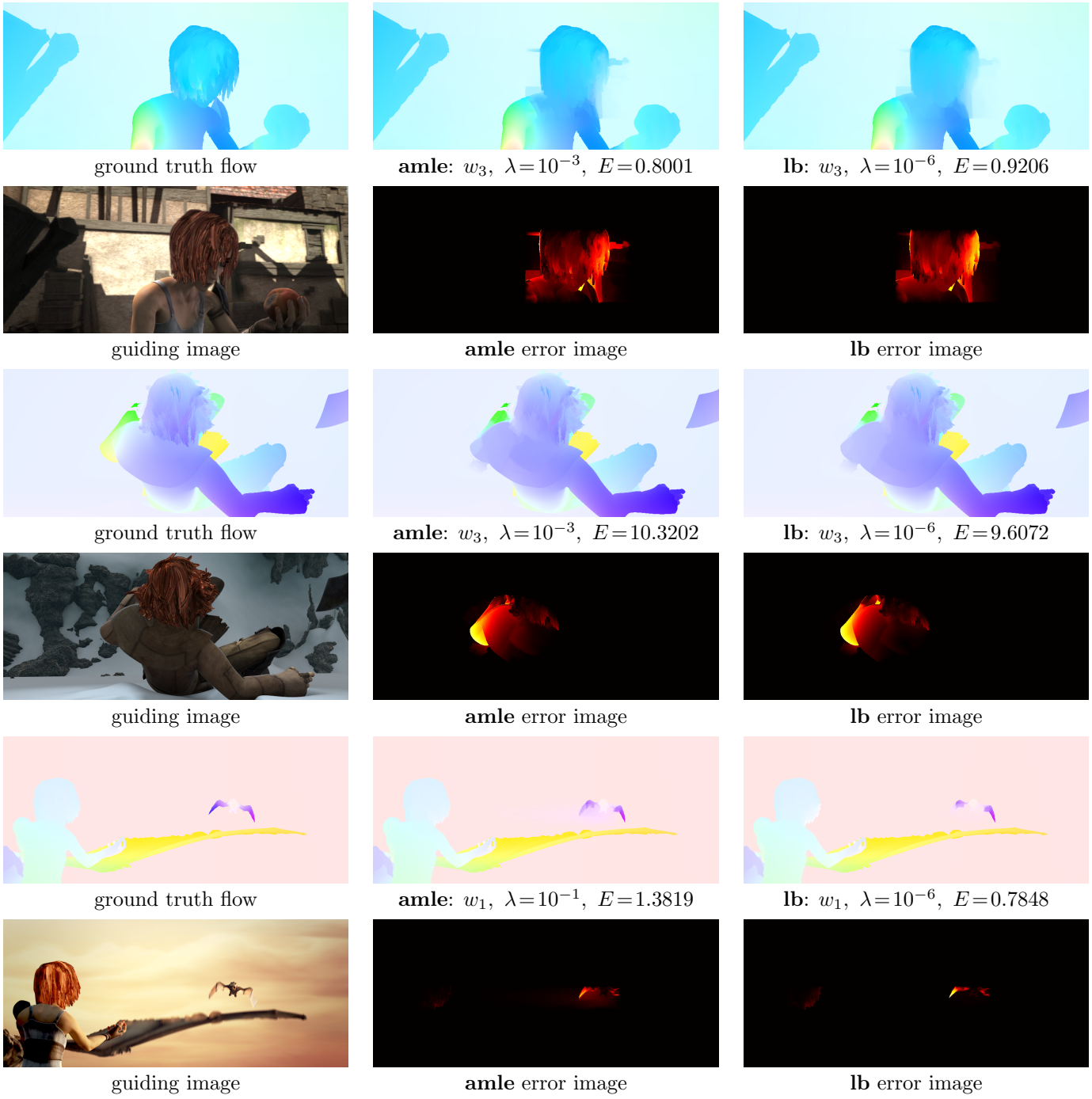

Figure 19: Inpainting of large holes using the masks in Figure 18. First, third and fifth row, from left to right: ground truth flow, geodesic AMLE inpainting result, LB inpainting result. Second, fourth and sixth row, from left to right: guiding image, geodesic AMLE and LB error images.

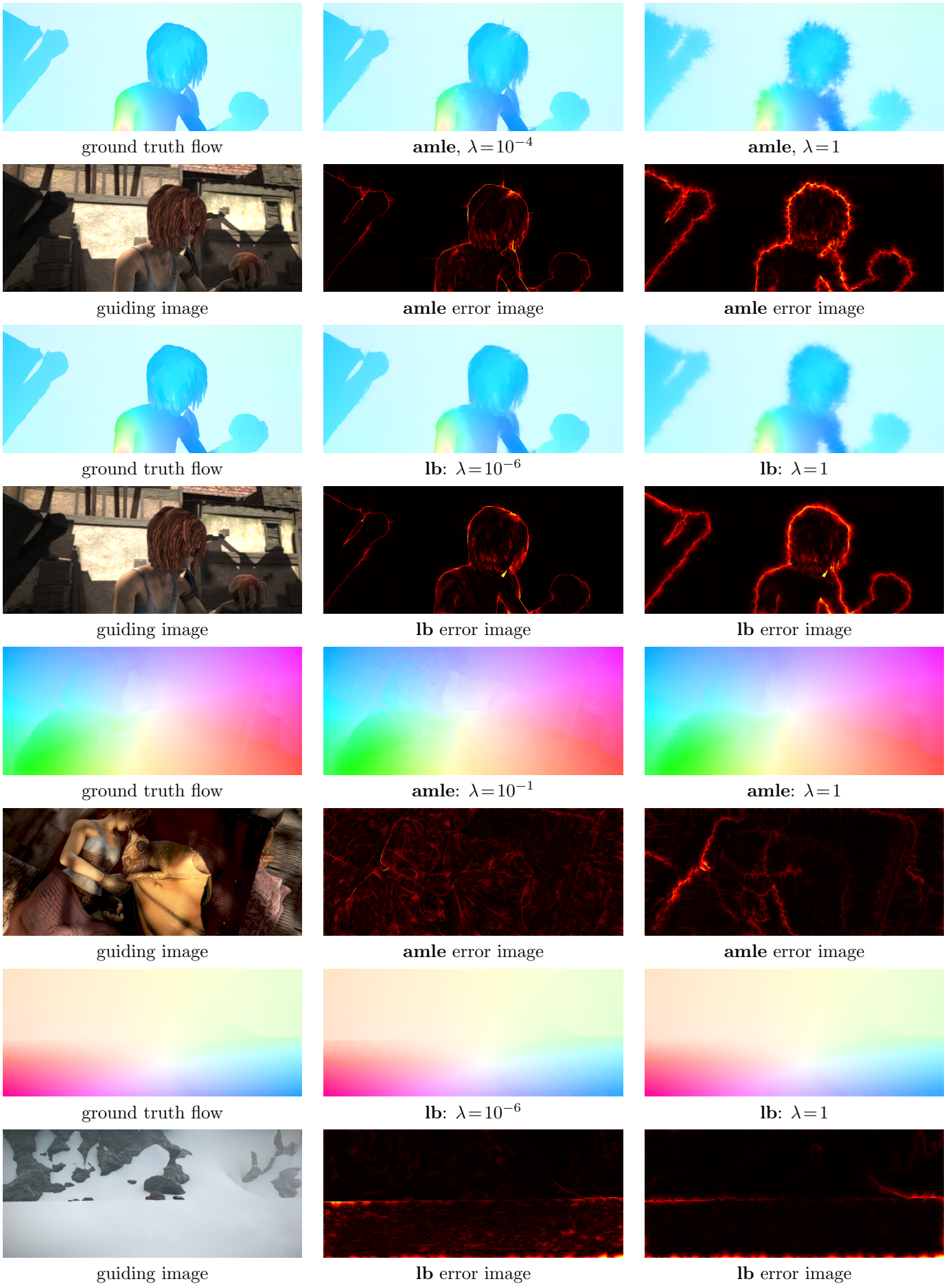

Figure 20: Comparing guided ($\lambda < 1$) and unguided results ($\lambda = 1$).

Acknowledgment

The first, third and fourth authors acknowledge partial support by MICINN/FEDER UE project, reference PGC2018-098625-B-I00 and by H2020-MSCA-RISE-2017 project, reference 777826 NoMADS.

Image Credits

All the images have been extracted from the MPI-Sintel Dataset (http://sintel.is.tue.mpg.de/downloads).

References

[1] A. Almansa, F. Cao, Y. Gousseau, and B. Rougé, *Interpolation of digital elevation models using AMLE and related methods*, IEEE Trans. Geosci. Remote Sens., 40 (2002), pp. 314–325. https://doi.org/10.1109/36.992791.

[2] G. Aronsson, *Extension of functions satisfying Lipschitz conditions*, Arkiv för Matematik, 6 (1967), pp. 551–561.

[3] G Aronsson, *On the partial differential equation* $u_x^2 u_{xx} + 2u_x u_y u_{xy} + u_y^2 u_{yy} 2 = 0$, Arkiv för Matematik, (1968).

[4] G. Aronsson, M. Crandall, and P. Juutinen, *A tour of the theory of absolutely minimizing functions*, Bulletin of the American Mathematical Society, (2004), pp. 439–505. https://doi.org/10.1090/S0273-0979-04-01035-3.

[5] R. Bergmann and D. Tenbrinck, *Nonlocal inpainting of manifold-valued data on finite weighted graphs*, in International Conference on Geometric Science of Information, Springer, 2017, pp. 604–612. https://doi.org/10.1007/978-3-319-68445-1_70.

[6] D. J. Butler, J. Wulff, G. B. Stanley, and M. J. Black, *A naturalistic open source movie for optical flow evaluation*, in European Conference on Computer Vision (ECCV), 2012. https://doi.org/10.1007/978-3-642-33783-3_44.

[7] V. Caselles, L. Igual, and O. Sander, *An axiomatic approach to scalar data interpolation on surfaces*, Numerische Mathematik, (2006), pp. 383–411. https://doi.org/10.1007/s00211-005-0656-8.

[8] V. Caselles, J-M. Morel, and C. Sbert, *An axiomatic approach to image interpolation*, IEEE Transactions on Image Processing, 7 (1998), pp. 376–386. https://doi.org/10.1109/83.661188.

[9] A. Elmoataz, M. Toutain, and D. Tenbrinck, *On the p-Laplacian and* ∞*-Laplacian on graphs with applications in image and data processing*, SIAM Journal on Imaging Sciences, 8 (2015), pp. 2412–2451. https://doi.org/10.1137/15M1022793.

[10] V. Fedorov, P. Arias, R. Sadek, G. Facciolo, and C. Ballester, *Linear multiscale analysis of similarities between images on Riemannian manifolds: Practical formula and affine covariant metrics*, SIAM Journal on Imaging Sciences, 8 (2015), pp. 2021–2069. https://doi.org/10.1137/141000002.

[11] L.J. Grady and J.R. Polimeni, *Discrete calculus: Applied analysis on graphs for computational science*, Springer Science & Business Media, 2010.

[12] R. Jensen, *Uniqueness of Lipschitz extensions: minimizing the sup norm of the gradient*, Archive for Rational Mechanics and Analysis, (1993), pp. 51–74. https://doi.org/10.1007/BF00386368.

[13] V. Lazcano, *Some Problems in Depth Enhanced Video Processing*, 2016. PhD. Thesis.

[14] O. Lézoray and L. Grady, *Image processing and analysis with graphs: theory and practice*, CRC Press, 2012.

[15] M. Mainberger, A. Bruhn, J. Weickert, and S. Forchhammer, *Edge-based compression of cartoon-like images with homogeneous diffusion*, Pattern Recognition, 44 (2011), pp. 1859–1873. https://doi.org/10.1016/j.patcog.2010.08.004.

[16] J.J. Manfredi, A.M. Oberman, and A.P. Sviridov, *Nonlinear elliptic partial differential equations and p-harmonic functions on graphs*, Differential Integral Equations, 28 (2015), pp. 79–102.

[17] A. Oberman, *A convergent difference scheme for the infinity Laplacian: construction of absolutely minimizing Lipschitz extensions*, Mathematics of Computation, 74 (2005). https://doi.org/10.1090/S0025-5718-04-01688-6.

[18] M. Oliver, L. Raad, C. Ballester, and G Haro, *Motion inpainting by an image-based geodesic AMLE method*, in 25th IEEE International Conference on Image Processing (ICIP), IEEE, 2018, pp. 2267–2271. https://doi.org/10.1109/ICIP.2018.8451851.

[19] O. Sander, V. Caselles, and M. Bertalmio, *Axiomatic scalar data interpolation on manifolds*, in IEEE International Conference on Image Processing, vol. 3, 2003, pp. III–681. https://doi.org/10.1109/ICIP.2003.1247336.