Chapter 1

Frame Field optimization

Until now, we have only covered how we measure the Dirichlet energy $||\mathcal{D}R||$ in the new metric g. This chapter covers how we minimize

$$E(\mathcal{M}) = \int_{\mathcal{M}} ||DR||^2.$$

We use an optimization scheme based on the PhD work of Simone Raimondi[1], which in turn is based on the Merriman-Bence-Osher (MBO) algorithm. In principle, any frame field optimization scheme that works based on optimizing the Dirichlet energy (or its discretized version $||R(q) - R(p)||^2$) can be modified with the rotation coefficient R_{qp} to optimize in a new metric.

1.1 Optimization Algorithm

The algorithm works by dividing the manifold into cubes, and optimizing the frames per cube. The algorithm works based on an adaptive grid, refining the grid and optimizing where more resolution is needed, see Figure 1.1 From a given voxel with coordinates $v = (x, y, z)$, the rotation coefficients are

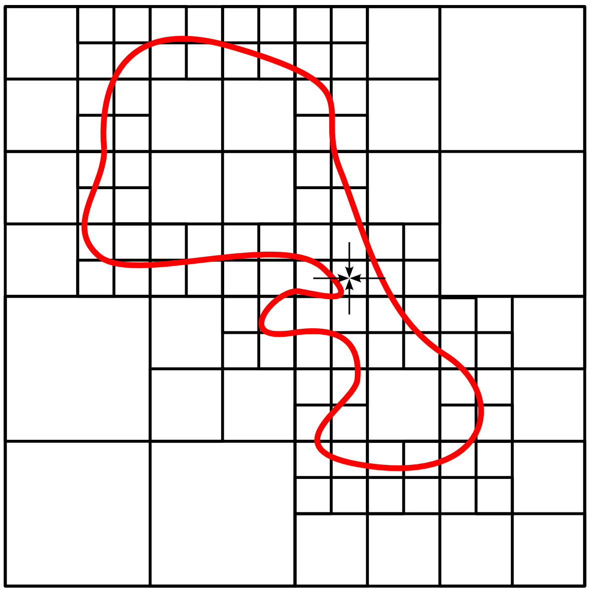

Figure 1.1. An adaptive grid is formed around the manifold (bold red), with increased resolution where needed. The rotation coefficients for a given voxel are calculated from its neighbours (arrows). In 3D, the cubes get divided into eight smaller cubes.

[1] `https://cgg.unibe.ch/person/1105/ - Simone Raimondi`

calculated from their neighbors $v_n = (x \pm 1, y \pm 1, z \pm 1)$ to v, i.e.

$$||[R_{v_n \rightsquigarrow v}]v_n - v||^2.$$

The MBO algorithm is based on repeated diffusion in the spherical harmonics (as averaging frames makes sense in the spherical harmonics), with a projection back to the manifold of valid frames (see Figure 1.2).

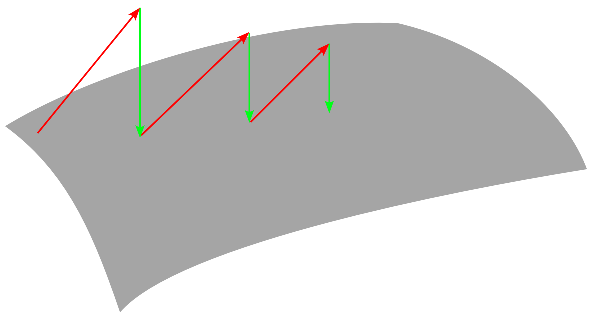

Figure 1.2. Repeated diffusion of frames in spherical harmonics with a projection back to the valid space of frames is used to minimize the energy. Figure inspired from [**?**]

Because the implementation for the adaptivity of the grid was not finished at the time of this thesis, the optimization is done on a regular grid where all voxels have the same size. Two parameters depth d and number of diffusion steps n are present. The depth d is how many times the grid gets subdivided, with depth 0 corresponding to a cube that is subdivided once. The resulting frame fields are expected to be the same, it just does it slower.

1.2 Caching of coefficients

Without any optimization, the algorithms time complexity is exponential with the depth. At depth n, there are 8^{n+1} voxels making up the enclosing cube. For every voxel, 6 rotation coefficients are calculated from its neighbors to itself and diffused d times. Because the underlying metric field does not change while optimizing, we can cache these rotation coefficients to not recalculate R from scratch at every diffusion step. We initialize a list of length $8^{n+1} \cdot 6$ at depth n, which in the first run at each depth gets filled with the corresponding rotation coefficients. Afterwards, the corresponding rotation coefficient for the voxel can get looked up at each diffusion step.