A Addtional Results

A.1 Local Observer Extraction

In our framework, spatial filtering methods include, but are not limited to, filtering based on proximity to isosurfaces (inside, outside, or close to isosurface boundaries) and using a dynamically moving bounding box defined by clusters of pathlines.

Local observer extraction can be achieved by first performing global optimization, followed by spatial filtering to identify the area of interest. However, a more efficient and practical alternative is to apply spatial filtering first, thus restricting numerical optimization exclusively to regions of interest when computing the local observer field.

After obtaining the local observer field, additional attribute-based filters can be applied to exclude pathlines exhibiting numerical instability or lacking smoothness. Users can then interactively select specific observer trajectories from the local observer field and fine-tune these trajectories to further enhance the visualization of targeted flow features.

In Fig. 11, we demonstrate the selection of local regions of interest using moving bounding boxes (see Sec. 6.3), along with the corresponding locally optimized observer field represented by vector glyphs.

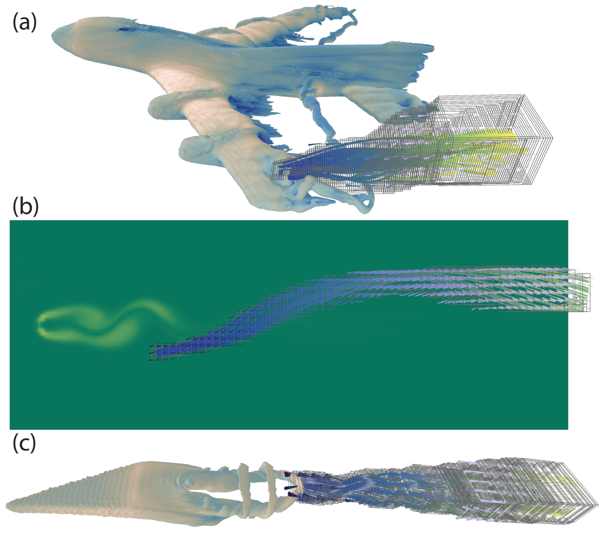

Fig. 11: Visualization of grid cells selected for the local optimization of reference frame transformations (timings reported in Table 3). We illustrate the pathline clusters along with their constructed time-dependent bounding boxes. The resulting observer field obtained from this local optimization is depicted using vector glyphs.

Case Study - Smoke Buoyancy. This dataset showcases multiple vortex ring structures that display deformation and irregular motion. The dataset spans a spatial extent of ([0, 5.88] × [0, 11.88] × [0, 5.88]) with a grid resolution of ([47, 95, 47]). Its time domain is ([0, 5.33]), comprising 160 timesteps. Fig. 12 effectively illustrates our method's ability to focus on a localized region within this dataset.

In Fig. 12(a), we define two sets of pathlines of interest in the lab-frame, highlighted by blue and yellow bounding boxes. We concentrate on the pathlines within the blue-highlighted region. For these, we construct a moving bounding box and then perform local optimization inside this box to derive an observer field that is defined exclusively within this moving domain. Fig. 12(b) visualizes the resulting local observer as a curve, along with its moving frame's X, Y, and Z axes. These axes are rendered as pink, light green, and light blue segments, respectively, which, when propagated along the worldline, define three corresponding surfaces of the same colors. In Fig. 12(c), applying this local observer to the pathlines reveals well-defined vertical motion for the blue-highlighted region. Conversely, this observer is entirely unsuitable for the yellow-highlighted pathlines, transforming them into a collection of chaotically behaving pathlines. This behavior of our method is expected as different regions of the dataset are best visualized relative to different observers. Finally, in Fig. 12(d), we directly apply our proposed "streamline-like" pathline filtering technique to the observed pathlines from (c). This process effectively filters out the pathlines within the yellow-highlighted box and the more peripheral pathlines within the blue-highlighted box. As a result, we obtain a visually clear depiction of particle trajectories that distinctly revolve around the vortex coreline in an appropriate local reference frame. By continuously adapting our focus to different segments of the coreline, we can consistently observe pathlines exhibiting localized vortical motion around the corresponding vortex ring segment.

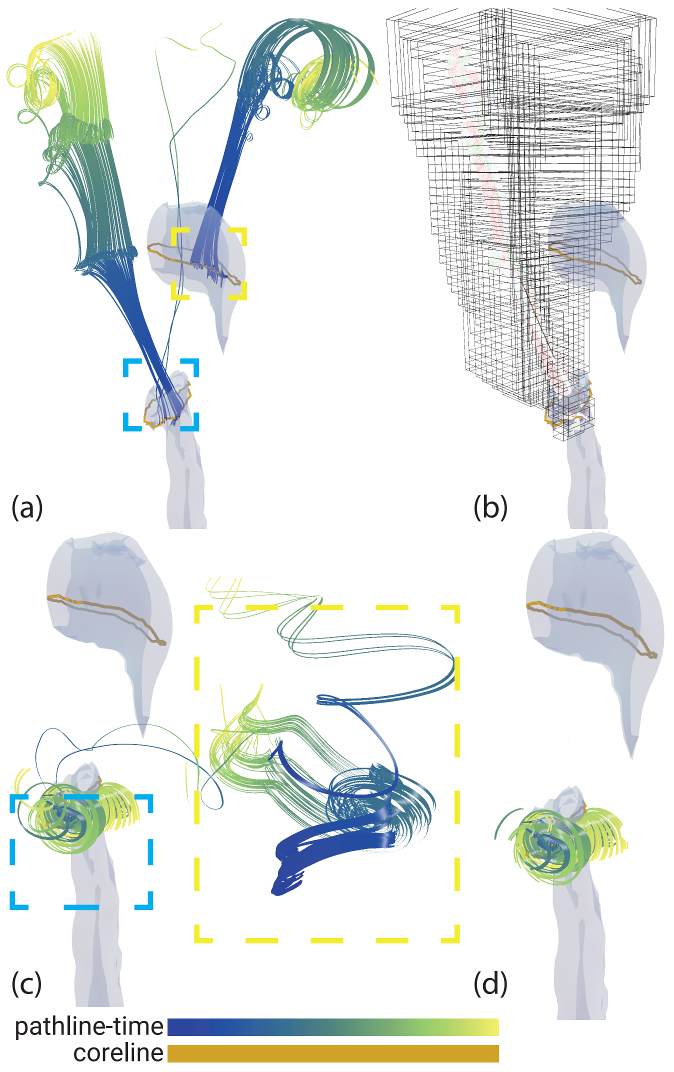

Fig. 12: Visualizing the **Smoke Buoyancy** dataset. We visualize a semi-transparent isosurface of vorticity magnitude together with corelines as yellow curves. (a) illustrates two sets of pathlines in the lab frame (highlighted in blue and yellow boxes). (b) shows the optimized local observer within a moving bounding box (light blue and pink frames) for the blue-highlighted pathlines. (c) presents the observed pathlines relative to the observer from (b). (d) shows the result of our "streamline-like" pathline filtering applied to (c).

A.2 Observed Streamlines and Streamline Filtering

In Figure 13 we show an example of observed streamlines. Observed streamlines are an example of the class of visual representations that need to be re-computed when we change the reference frame (see Section 6 - *Spatial primitives that experience general transformations*). Figure 13 shows different frames from the animation with streamlines in the lab-reference frame (Fig. 13 (bottom row)) as well as relative to a co-moving reference frame (Fig. 13 (top row)). The observer-relative streamlines reveal the features in the flow and align with the

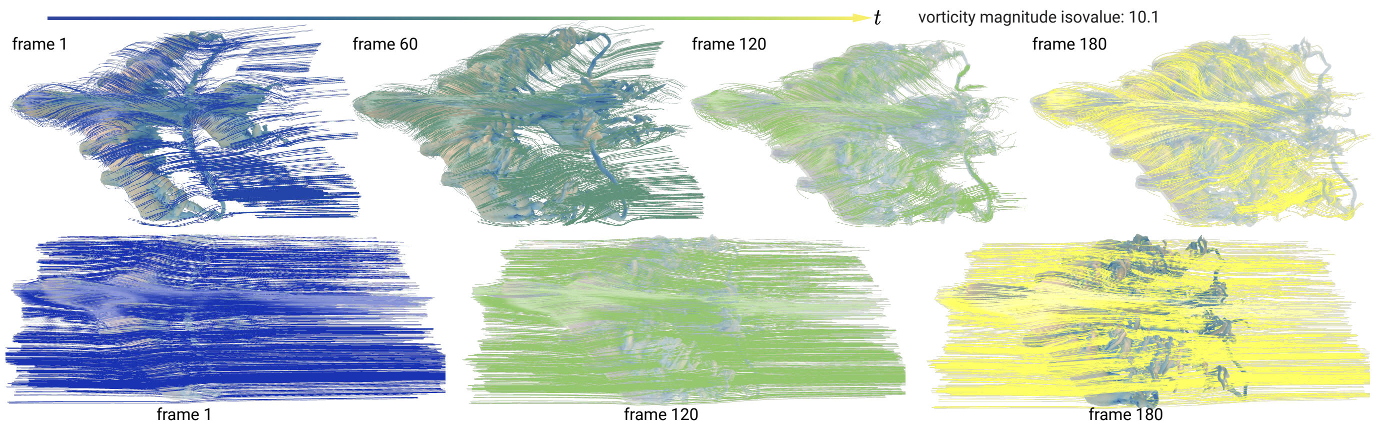

Fig. 13: **Comparison of observed and lab-frame streamlines in the Boeing 747 dataset.** Bottom row: Frames of an animation of streamlines in the lab frame using identical seeding positions and integration parameters. Top row: Frames of an animation of observer-relative streamlines. The observer-relative streamlines significantly reduce visual clutter and reveal features in the fluid flow more clearly at each timestep.

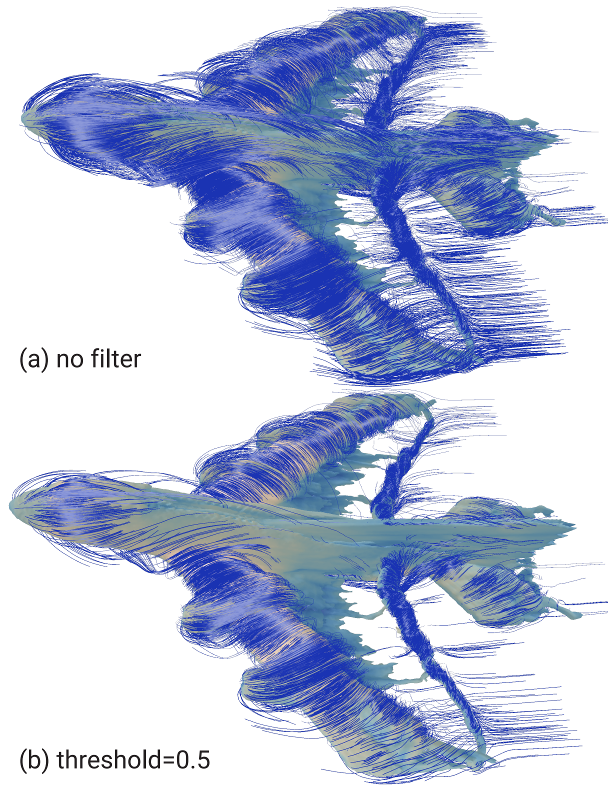

Fig. 14: **Observed streamlines in the Boeing 747 dataset with "steadiness" filter.** (a) Observed streamlines without filtering. (b) Observed streamlines with filter. Filtered streamlines further reduce visual clutter and reveal features in the fluid flow that are near steady over a local temporal neighborhood.

iso-surfaces of vorticity magnitude also shown in the animation.

In Figure 14 we show the effect of our "steadiness" filter applied to streamlines. In Figure 14 (a) we show observed streamlines, while in Figure 14 (b) we show the result of filtering streamlines by their similarity to pathlines. The streamlines that are similar to pathlines in the co-moving reference frame are part of regions that stay coherent over time.

B Implementation Details

Our experiments were conducted on a machine equipped with two Intel(R) Xeon(R) Gold 6230R CPUs running at 2.10GHz. Our framework is implemented in C++20 with OpenGL for rendering. We leverage OpenMP for CPU-side parallelism. We include pseudocode for our observer-relative pathline filtering algorithm in Algorithm 1, and present the pseudocode for the observer-aware iso-surface animation in Algorithm 2. Code for *Observer-Relative Scalar Field Transformation*, *Observer-Relative Pathline Filtering* and *Observer-Relative Isosurface Animation* is available at GitHub: https://github.com/CindyxdZhang/PyflowVis.

C Observer Comparison

We define an inner product of vector fields

$$\langle\langle \mathbf{w}_1, \mathbf{w}_2 \rangle\rangle = \int_U \langle \mathbf{w}_1(x), \mathbf{w}_2(x) \rangle \, dV, \tag{21}$$

where $U \subset \mathbb{R}^3$ is a chosen region and $\langle \cdot, \cdot \rangle$ denotes the usual Euclidean dot product at each point x. However, because $\mathfrak{se}(3)$ is a finite-dimensional vector space of dimension six, and therefore each $\mathbf{w}_i$ can be expanded in the $\{\mathbf{e}_i\}$ basis, we can compute the inner product above simply via a 6×6 matrix multiplication of coefficient vectors.

Observer Similarity. Given two time-dependent observers $\mathbf{w}_1$ and $\mathbf{w}_2$, we can measure their *difference* over a time interval $[t_0, t_1]$ via

$$d(\mathbf{w}_1, \mathbf{w}_2) = \int_{t_0}^{t_1} \sqrt{\langle\langle \mathbf{w}_1(\cdot,t) - \mathbf{w}_2(\cdot,t), \mathbf{w}_1(\cdot,t) - \mathbf{w}_2(\cdot,t) \rangle\rangle} \, dt. \tag{22}$$

D Reconstructing a Killing Observer Field

We can **reconstruct an entire 3D Killing field w** by knowing only two pieces of information at each point $p(t)$ along a trajectory $t \to p(t)$:

1. A single vector $\mathbf{w}(p(t), t)$.
2. A 3×3 skew-symmetric matrix $(\nabla \mathbf{w})(p(t), t)$, which inherently possesses only three degrees of freedom.

This reconstructive capability stems from the fundamental nature of Killing fields in Euclidean space. For such fields, the gradient $\nabla \mathbf{w}$ is not only skew-symmetric but also **constant throughout the entire space**. This means that if we know $(\nabla \mathbf{w})$ at one point, we effectively know it everywhere. Following Eq. 6, we decompose the 3D Killing field $\mathbf{w}(x,t)$ into a combination of translational and rotational components. An important observation is that the coefficients $q_1(t), \ldots, q_6(t)$ (which represent the magnitudes of the translational and rotational components) do **not depend on the position** p. Similarly, the gradient $\nabla w(t)$ is formed from $\mathbf{e}_4, \mathbf{e}_5, \mathbf{e}_6$) and also does **not depend on position** p.

At a given point p and time t, the Killing field $\mathbf{w}(p,t)$ can be expressed as:

$$\mathbf{w}(p,t) = q_1(t)\mathbf{e}_1 + q_2(t)\mathbf{e}_2 + q_3(t)\mathbf{e}_3 + (\nabla \mathbf{w}(t))p$$

Algorithm 1 Observer Relative Pathline Filtering

Input:

- $\mathscr{S}_0 = \{$pre-integrated pathlines using RK4$\}$
- Observer $\phi(t)$: time-dependent transformation stored as sequence of matrices
- Input vector field $v(x,t)$ and observer field $w(x,t)$
- Parameters: step size δ, segment length L, skip interval k, thresholds τ, ε

Begin:

```
Set maxIterations $M_I \leftarrow L/\delta$
 1: // Step 1: Compute observer-relative pathlines
 2: for each pathline $p_{\text{raw}} \in \mathscr{S}_0$ do
 3:     Initialize $p_{obs} \leftarrow$ empty list
 4:     for each point $(\mathbf{x},t) \in p_{\text{raw}}$ do
 5:         $\mathbf{x}_{\text{obs}} \leftarrow \phi(t) \cdot \mathbf{x}$
 6:         Append $(\mathbf{x}_{\text{obs}}, t)$ to $p_{obs}$
 7:     end for
 8:     Add $p_{obs}$ to $\mathscr{S}_1$          ▷ $\mathscr{S}_1$ stores observer-relative pathlines
 9: end for

10: // Step 2: Filtering via streamline comparison
11: for each pathline $p_{\text{raw}} \in \mathscr{S}_0$ do
12:     $p_{\text{obs}} \leftarrow$ corresponding pathline in $\mathscr{S}_1$
13:     for every kth point $(\mathbf{x}_0, t_0) \in p_{\text{raw}}$ do
14:         Perform RK4 streamline integration (both directions $M_I$
            steps) from $(\mathbf{x}_0, t_0)$ in relative field $v(x,t) - w(x,t)$
15:         Let $s_{\text{rel}} \leftarrow$ resulting streamline
16:         Let $s_{\text{obs}} \leftarrow \phi(t_0) s_{\text{rel}}$
17:         Compute difference: $d \leftarrow \frac{\sum_{i=start}^{end} \|s^{\text{i}}_{\text{obs}} - p^{\text{i}}_{\text{obs}}\|}{M_I}$
18:         Compute time-derivative: $\frac{\partial v}{\partial t}$ at $(\mathbf{x}_0, t_0)$
19:         if $d > \tau$ or $\left\| \frac{\partial v}{\partial t} \right\| < \varepsilon$ then
20:             Discard point $(\mathbf{x}_0, t_0)$ from $p_{\text{raw}}$
21:         end if
22:     end for
23: end for
24: return updated $\mathscr{S}_0$
```

Algorithm 2 Observer-Aware Iso-Surface Animation

Input:

- Observer $\phi(t)$: time-dependent transformation stored as sequence of matrices
- Scalar field $S(\mathbf{x},t)$
- Isovalue σ, number of frames N, time range $[t_{\text{min}}, t_{\text{max}}]$

Begin:

```
Set $\Delta t \leftarrow (t_{\text{max}} - t_{\text{min}})/(N-1)$
 1: // Step 1: Transform scalar field into observer frame
 2: for each timestep index $i$ in $S$ do
 3:     $t \leftarrow$ time corresponding to index $i$
 4:     Compute $S_{\text{obs}}(\mathbf{x},t) \leftarrow S(\phi^{-1}(t) \cdot \mathbf{x}, t)$
 5: end for

 6: // Step 2: Marching cubes on observer-relative scalar field
 7: for frame index $j = 0$ to $N-1$ do
 8:     $t_c \leftarrow t_{\text{min}} + j \cdot \Delta t$
 9:     $S_{\text{obs}}(\mathbf{x},t)|_{t_c} \leftarrow$ temporal interpolation $S_{\text{obs}}(\mathbf{x},t)$
10:     Run marching cubes on $S_{\text{obs}}(\mathbf{x},t)|_{t_c}$ with isovalue $\sigma$
11:     Let $\mathscr{I}_j$ be the resulting iso-surface
12:     Optionally apply inverse transform: $\mathscr{I}^o_j \leftarrow \phi(t_c) \cdot \mathscr{I}_j$          ▷
    transform iso-surface back to lab frame
13: end for
14: return Observed Iso-surface sequence $\{\mathscr{I}_j\}_{j=0}^{N-1}$ or Iso-surface
    sequence in lab frame $\{\mathscr{I}^o_j\}_{j=0}^{N-1}$
```

From this, we can isolate the translational component:

$$q_1(t)\mathbf{e}_1 + q_2(t)\mathbf{e}_2 + q_3(t)\mathbf{e}_3 = \mathbf{w}(p,t) - (\nabla \mathbf{w}(t))p$$

Now, to reconstruct the Killing field $\mathbf{w}(x,t)$ at *any* arbitrary point x at time t, we can substitute this expression back into the general form:

$$\mathbf{w}(x,t) = (q_1(t)\mathbf{e}_1 + q_2(t)\mathbf{e}_2 + q_3(t)\mathbf{e}_3) + (\nabla \mathbf{w}(t))x$$

Replacing the bracketed translational terms:

$$\mathbf{w}(x,t) = (\mathbf{w}(p,t) - (\nabla \mathbf{w}(t))p) + (\nabla \mathbf{w}(t))\mathbf{x}$$

This simplifies to the concise reconstruction formula:

$$\mathbf{w}(x,t) = \mathbf{w}(p,t) + (\nabla \mathbf{w}(t))(x - p)$$

This equation demonstrates how the entire 3D Killing field $\mathbf{w}$ at any point x can be reconstructed given its value and gradient at a single observed point p. This property is fundamental to our Killing field model observer, enabling subsequent filtering, and interpolation of observers.