Image-Based Bidirectional Scene Reprojection

Lei Yang[1], Yu-Chiu Tse[1],
Pedro Sander[1], Jason Lawrence[2],
Diego Nehab[3,4], Hugues Hoppe[3],
Clara Wilkins[5]

1 香港科技大學 THE HONG KONG UNIVERSITY OF SCIENCE AND TECHNOLOGY

2 UNIVERSITY of VIRGINIA

3 Microsoft® Research

4 impa NATIONAL INSTITUTE FOR PURE AND APPLIED MATHEMATICS, BRAZIL

5 WESLEYAN UNIVERSITY

Outline

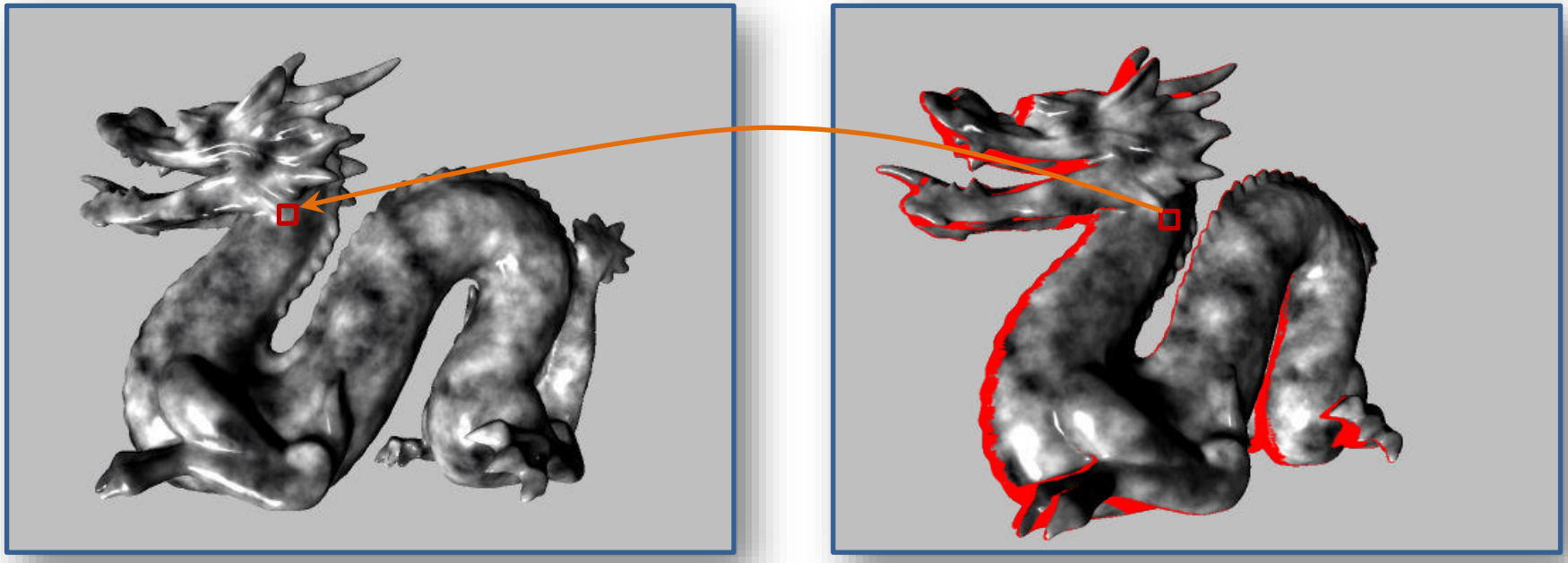
- Reprojection and data reuse
  - Taxonomy
- Bidirectional reprojection
  - Scene-assisted reprojection
  - Image-based reprojection
  - Interoperability
- Partitioned rendering and lag
  - User study
- Results

Goal

- Optimize performance for real-time rendering
  - For complex shading tasks
  - For low-end platform adaptation
- A general-purpose acceleration method
  - Generate in-between frames with low cost
  - In real-time (interactive)
  - Trade quality for performance

Reprojection for data reuse

- Generate in-between frames with low cost [Scherzer'11]:
  Reproject and reuse pixel data from similar frames
- Avoid redundant computation
- Newly disoccluded regions can be missing



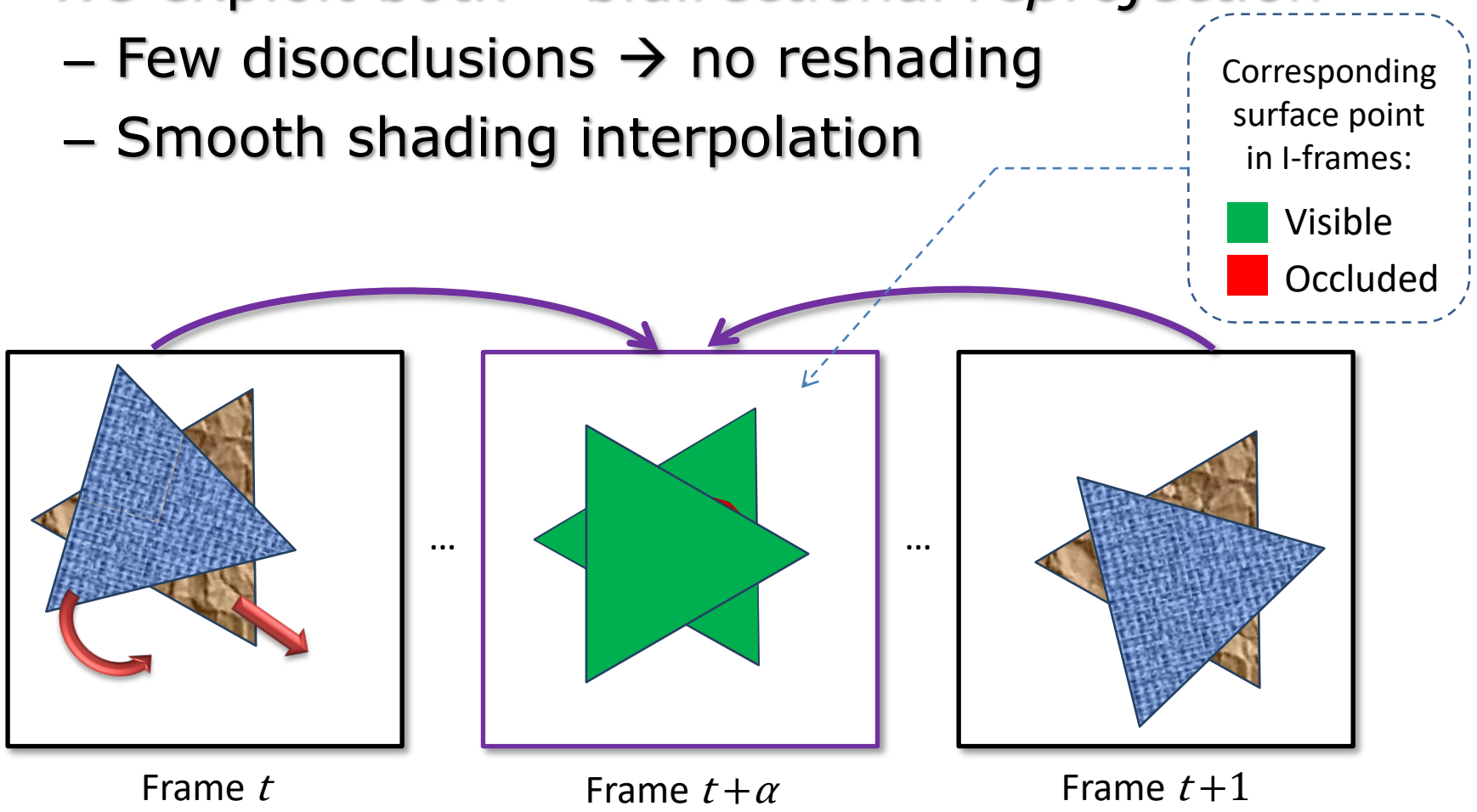
Frame t Frame $t+\alpha$

Reprojection for data reuse

- Taxonomy for reprojection methods
  - Temporal direction
  - Data access
  - Correspondence domain

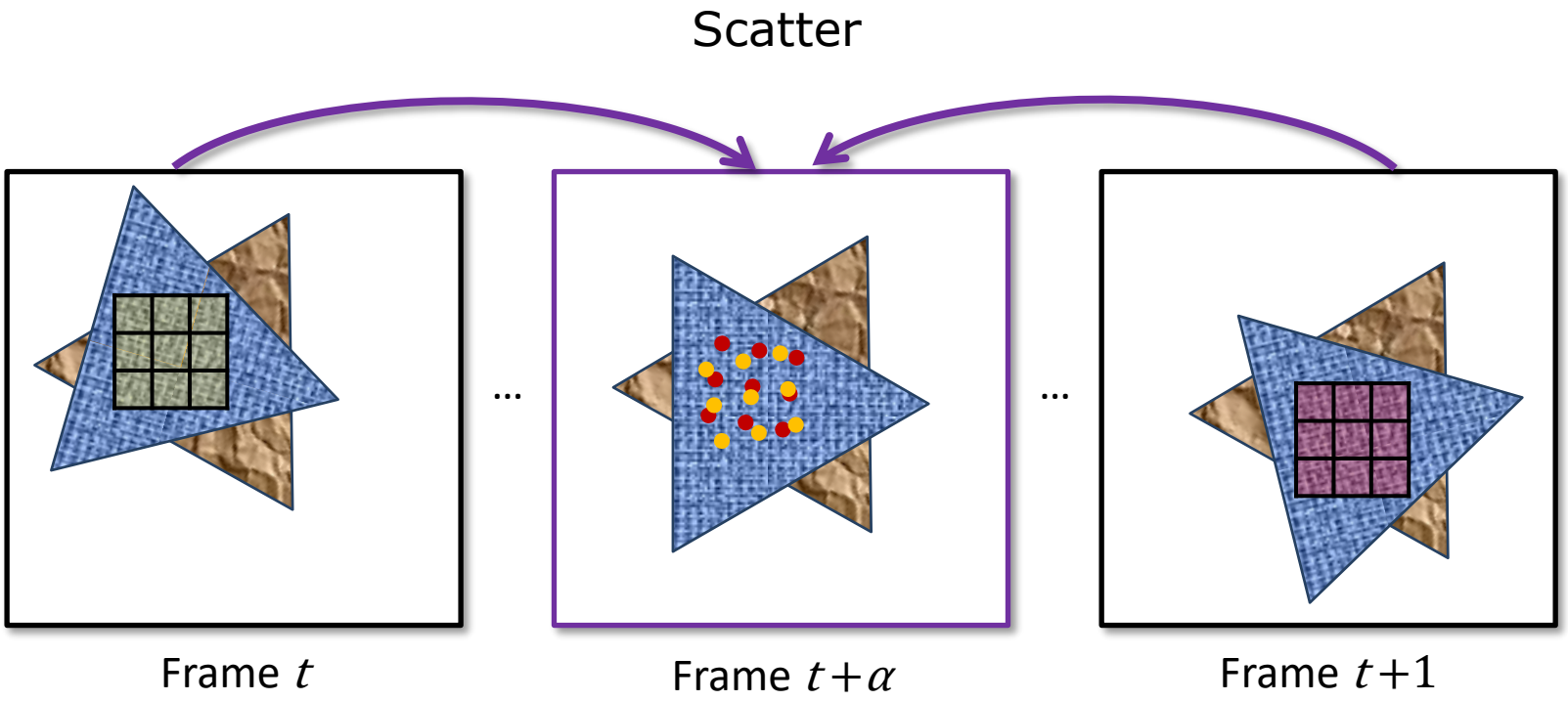
Reprojection: Temporal direction

- Forward vs. Backward
- We exploit both – *bidirectional reprojection*
  - Few disocclusions → no reshading
  - Smooth shading interpolation



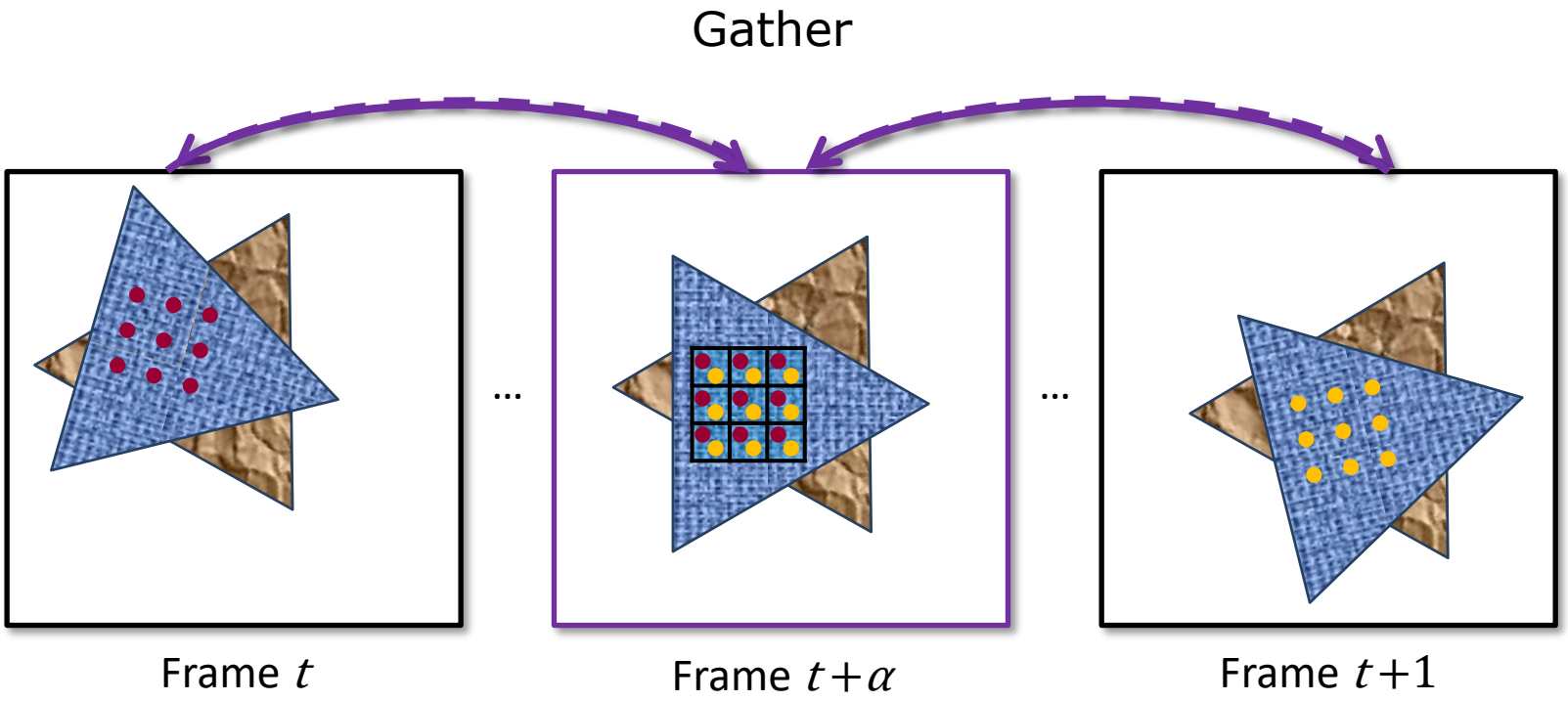
Reprojection: Data access

- Scatter vs. Gather



Reprojection: Data access

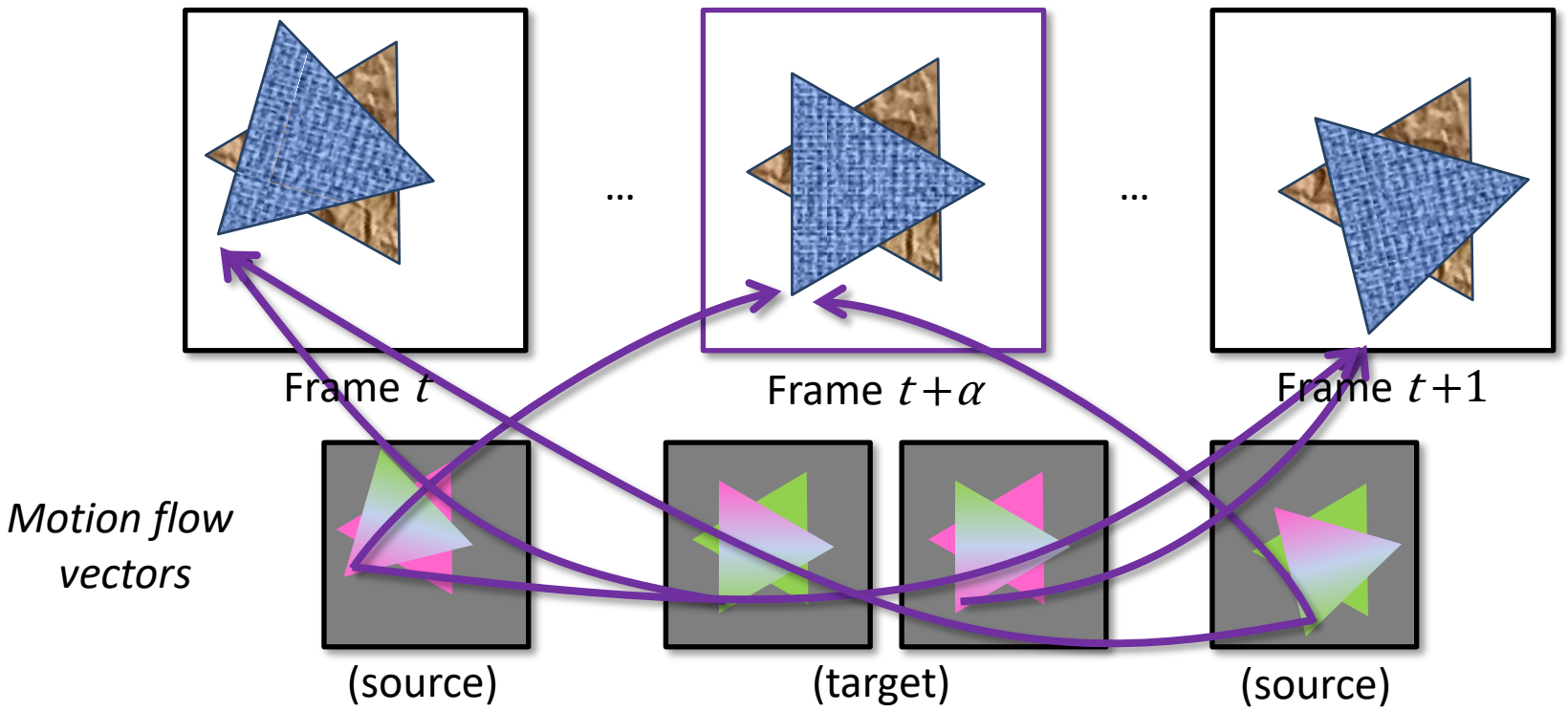
- Scatter vs. Gather
- We choose "gather"
  - Simpler, faster, higher quality filtering



Reprojection: Correspondence domain

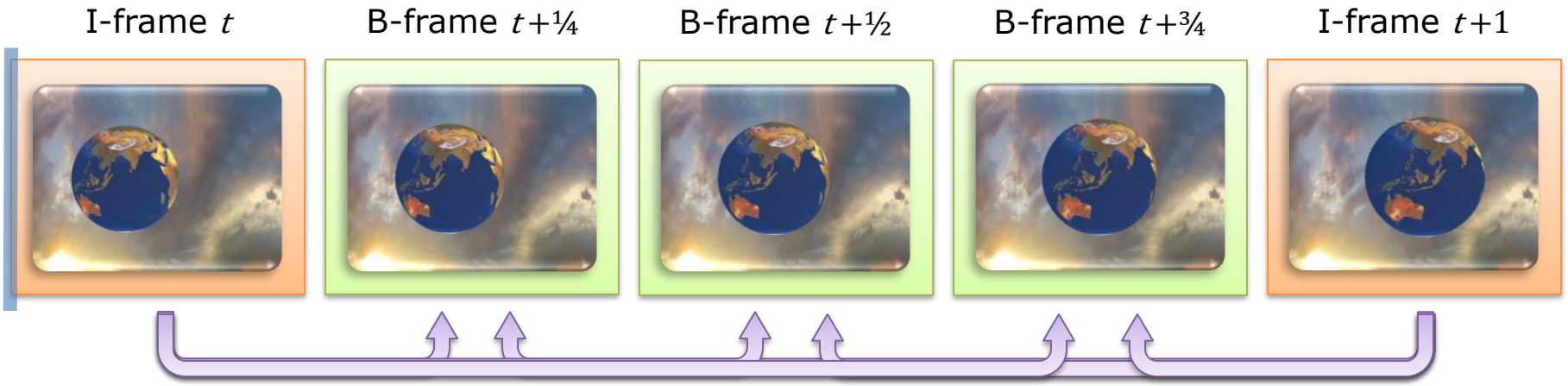
-- Domain where the motion flows are stored

- Source (w/ scatter) vs. target (w/ gather)
- We propose "source" (gather-based)



Overview of our approach

- Render *I-frames*, insert interpolated *B-frames*
- Use bidirectional reprojection ("*Bireproj*")
- Two approaches:
  - *Scene-assisted*: extension of [Nehab'07]
  - *Image-based*: main contribution



Scene-assisted Bireproj

- Rasterize each B-frame
  - Perform reprojection[Nehab'07] onto both I-frames
  - Occlusion test: reprojected depth = stored depth?
  - Blend visible results based on α

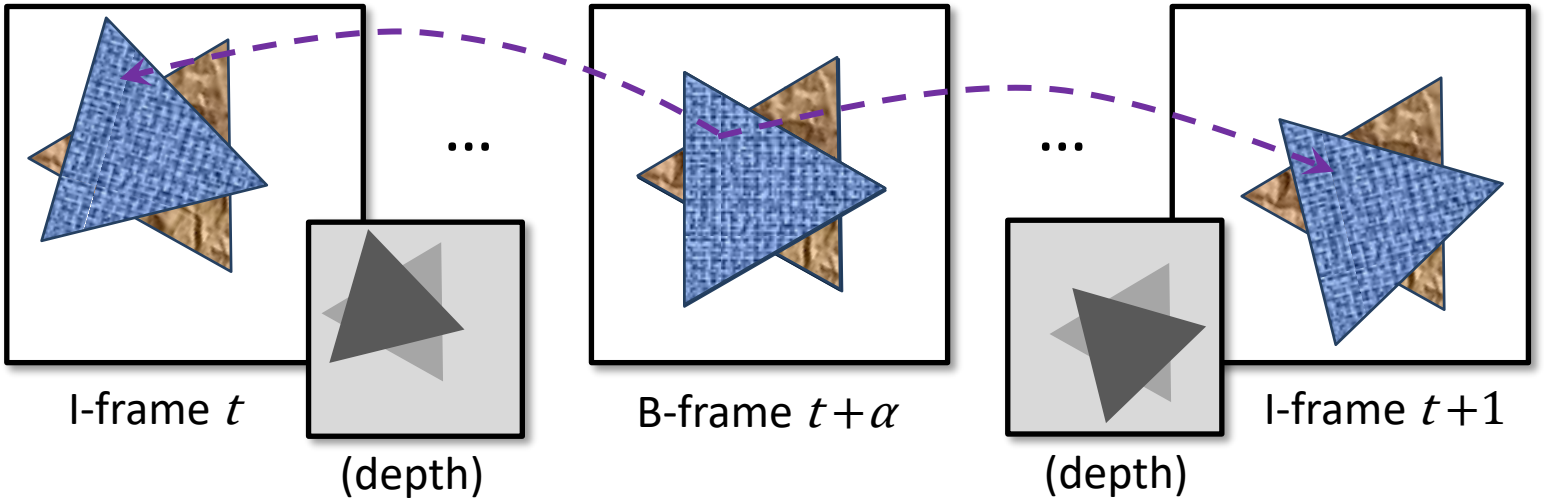
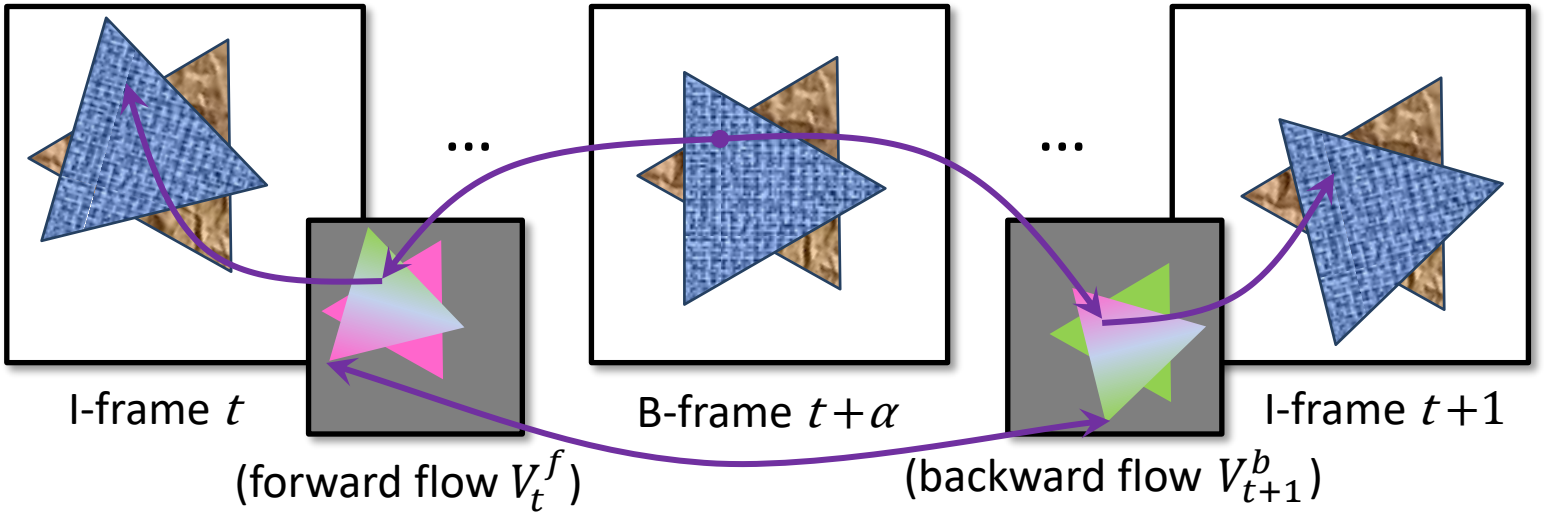


Image-based Bireproj

- Reprojection by searching in flow fields
  - Generate motion flow fields for each pair of I-frames
  - For each pixel in B-frame $t+\alpha$
    - Search in forward flow field V_t^f to reproject to I-frame t
    - Search in backward flow field V_{t+1}^b to reproject to I-frame $t+1$
    - Load and blend colors from frame t and $t+1$



The iterative search algorithm

- Assumptions:
  1. The motion between t and $t+1$ is linear
  2. The motion flow field is continuous and smooth
- Given $p_{t+\alpha}$, find p_t in field V_t^f such that

$$p_t + \alpha V_t^f[p_t] = p_{t+\alpha}$$

  - Same for p_{t+1} (in reverse)
- An inverse-mapping problem

Motion flow

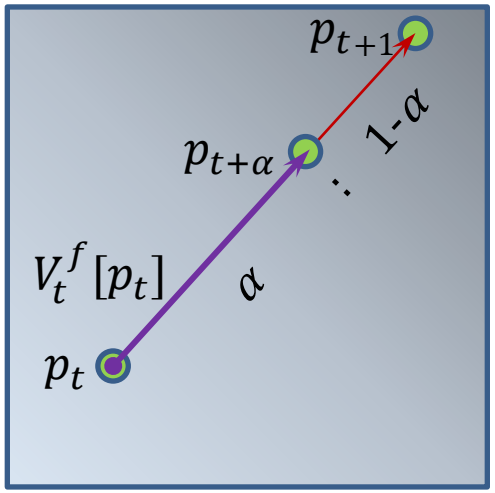
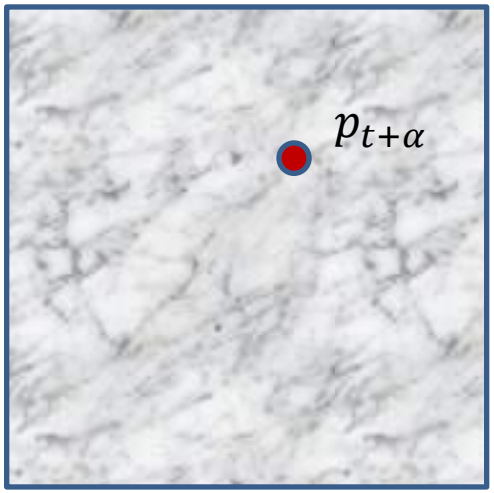


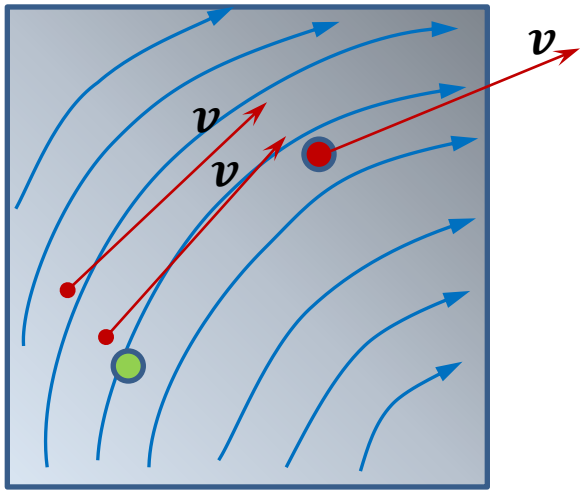
Image-space



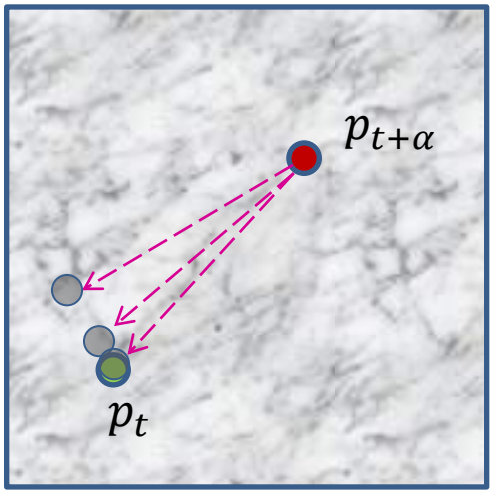
The iterative search algorithm

- Iterative search
  1. Initialize vector $\boldsymbol{v}$ with the motion flow $\alpha V_t^f[p_{t+\alpha}]$
  2. Attempt to find p_t using $\boldsymbol{v}$
  3. Update $\boldsymbol{v}$ with the motion flow at current p_t estimate
  4. Repeat 2-3 (3 iterations suffice in our experiments)

Motion flow

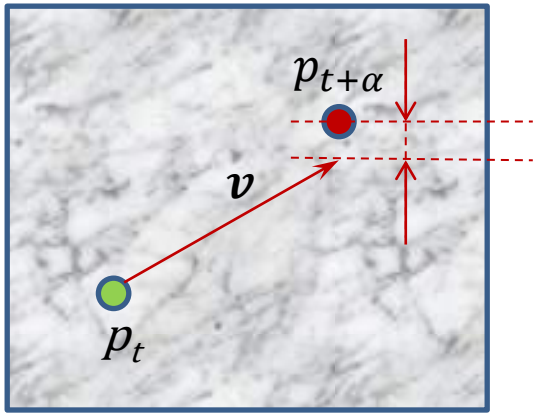


Iterative reprojection



Visibility test criteria

1. Screen-space reprojection error
   - Residual between $p_t + \boldsymbol{v}$ and $p_{t+\alpha}$
   - Large error → unreliable p_t
   - p_t & p_{t+1}: use the more precise side to readjust the other

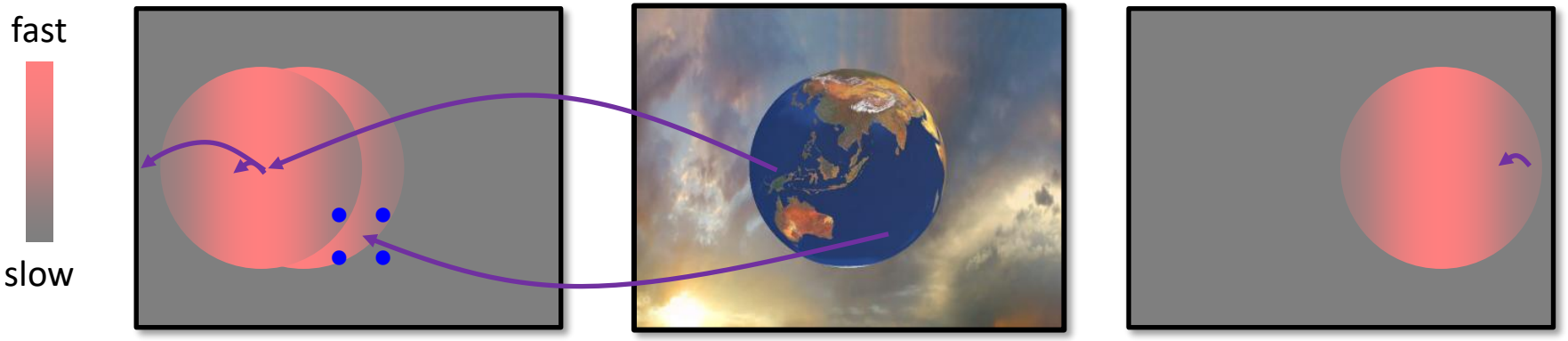


Reprojection error

2. Scene depth
   - Significantly different scene depths imply occlusion
   - Trust the closer one (smaller depth)

Additional search initialization

- The motion field is often only piecewise smooth
  a) Imprecise initial vector across object boundaries
  b) Search steps can fall off the object
- For a) :
  - Additional 4 candidates within a small neighborhood
  - Initialize using the result from a closer B-frame
- For b):
  - Initialize using the vector from the opposite I-frame

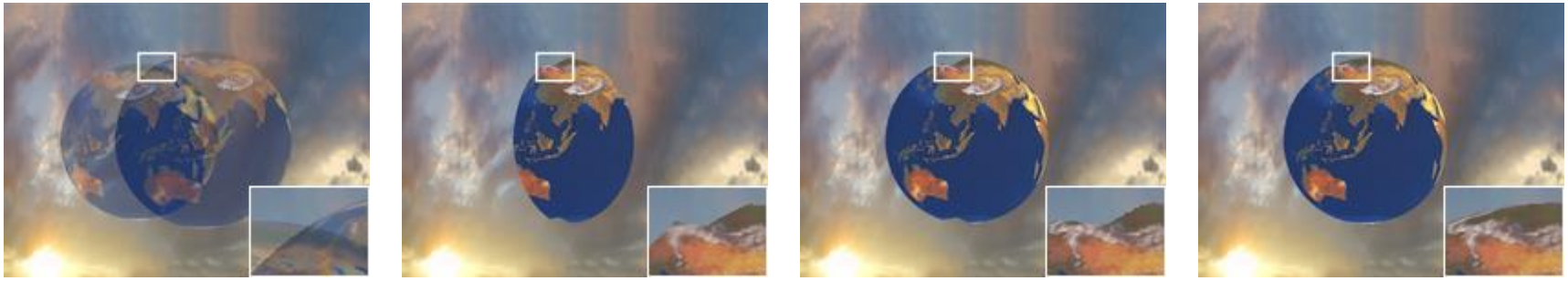


Additional search initialization

- Comparison

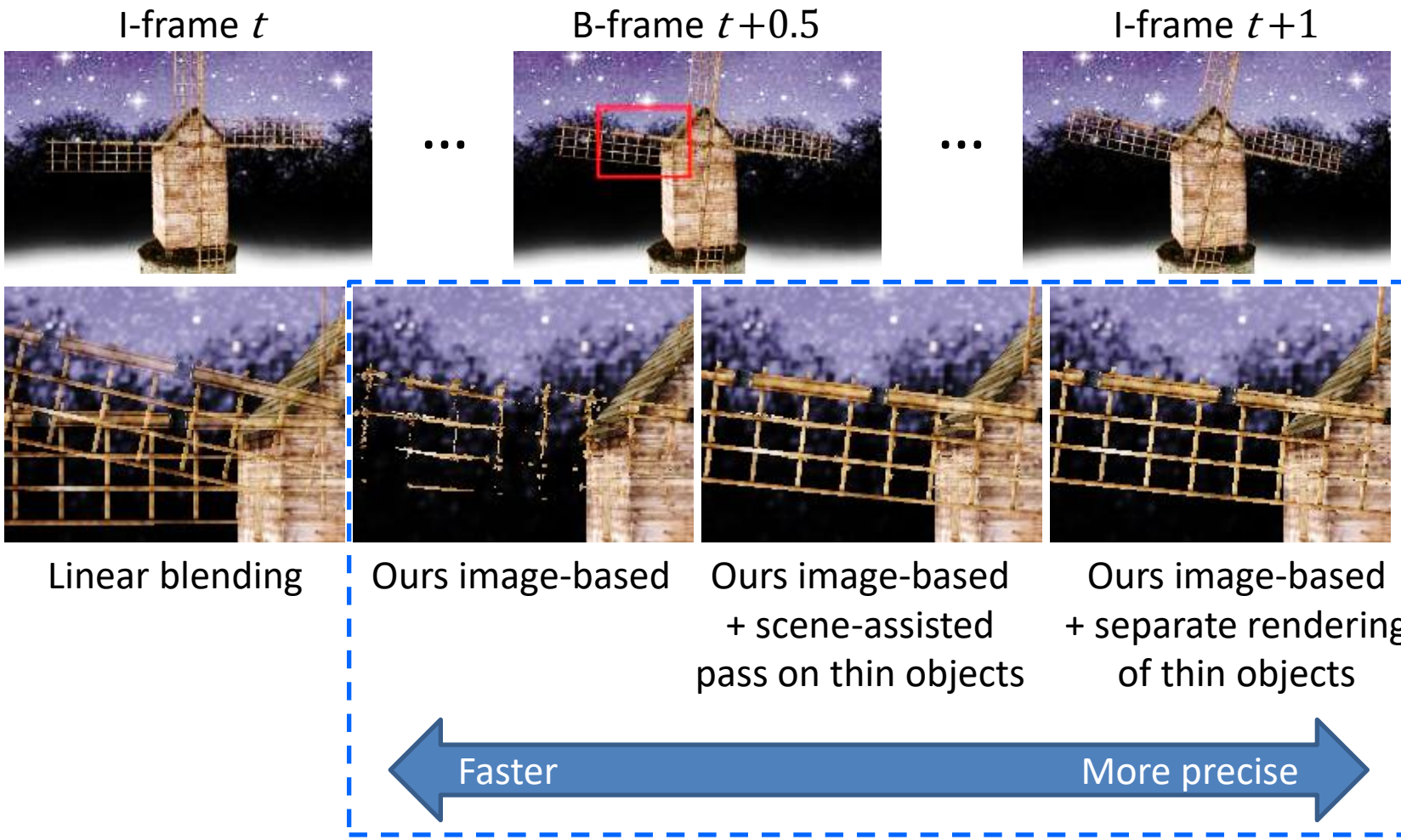
I-frame t | B-frame $t+¼$ | B-frame $t+½$ | B-frame $t+¾$ | I-frame $t+1$

Linear blending | Image-based (No additional init.) | Image-based (with “b”) | Image-based (with “a+b”)



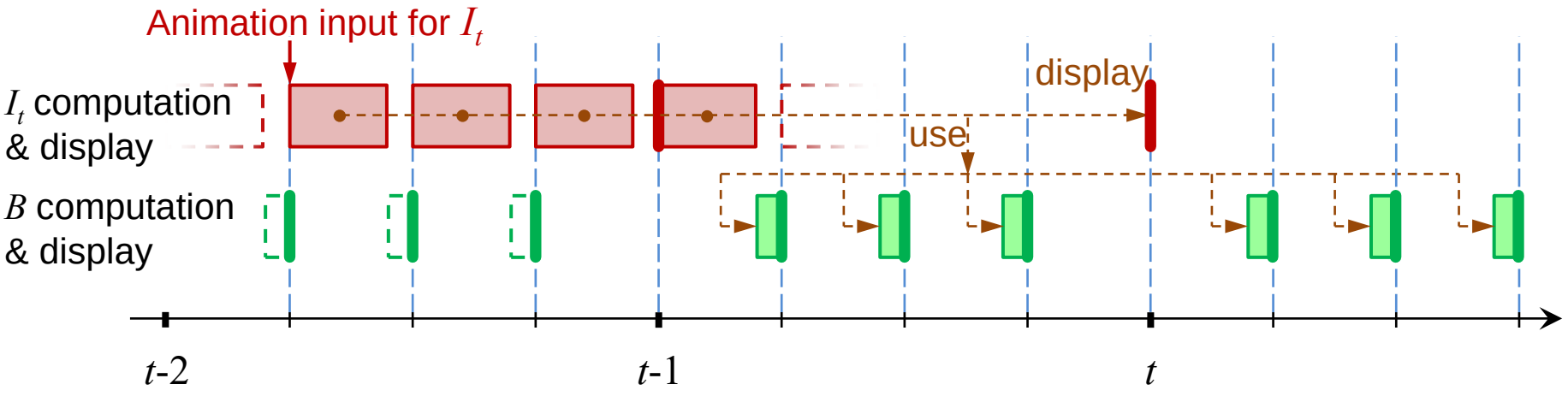
Interoperability

- Problem with fast moving thin objects
- Solution: mix multiple approaches (buffers shared)



Partitioned rendering

- I-frame shading parallel to B-frame generation
- Partition the I-frame rendering tasks evenly
  - Compute each group during a frame display
- No need to partition with (future) GPU multitasking
- I-frame "t" must start rendering at $t - 1 - \frac{n-1}{n}$
  - A potential lag



Lag

- Lag with standard double buffering:
  - Original: 1 time step (ts)
  - Bireproj: I-frame: $1 + \frac{n-1}{n}$ ts, B-frame: $1 + \frac{1}{n}$ ts
- Lag with 1-frame render ahead:
  - Original: 2 ts
  - Bireproj: 2 ts (I-frame)
- Conjecture:

  Lag with Bireproj is similar to the standard lag

User study

- The ball shooting game
  - Goal: click green balls, avoid red ones and null clicks
  - Subjects play in different modes and record results

User study

- Modes:
  - Standard rendering 15fps (Baseline)
  - Simulated 30fps / 60fps
  - Artificially lagged 50/100/200ms (on 60fps)

  to be compared against:

  - Bireproj (15 → 60fps)

User study

- Conclusions:
  - Our method did better than 15fps, but worse than 30fps
  - Perceived lag: 50ms < Bireproj << 100ms
    (The lag of standard 15fps is 66.7ms)

	Mode	60fps*	30fps*	15fps	Lag 50ms	Lag 100ms	Lag 200ms
objective	Green Hits	●	●	●	●	●	●
	Red Hits	●	●	●	●	●	●
	Misses	●	●	●	●	●	●
subjective	Enjoyment	●	●	●	●	●	●
	Difficulty	●	●	●	●	●	●
	Responsiveness	●	●	●	●	●	●
	Smoothness	●	●	●	●	●	●

● Better than Bireproj ● Worse than Bireproj ● No significant difference

* infeasible in real scenarios

Results

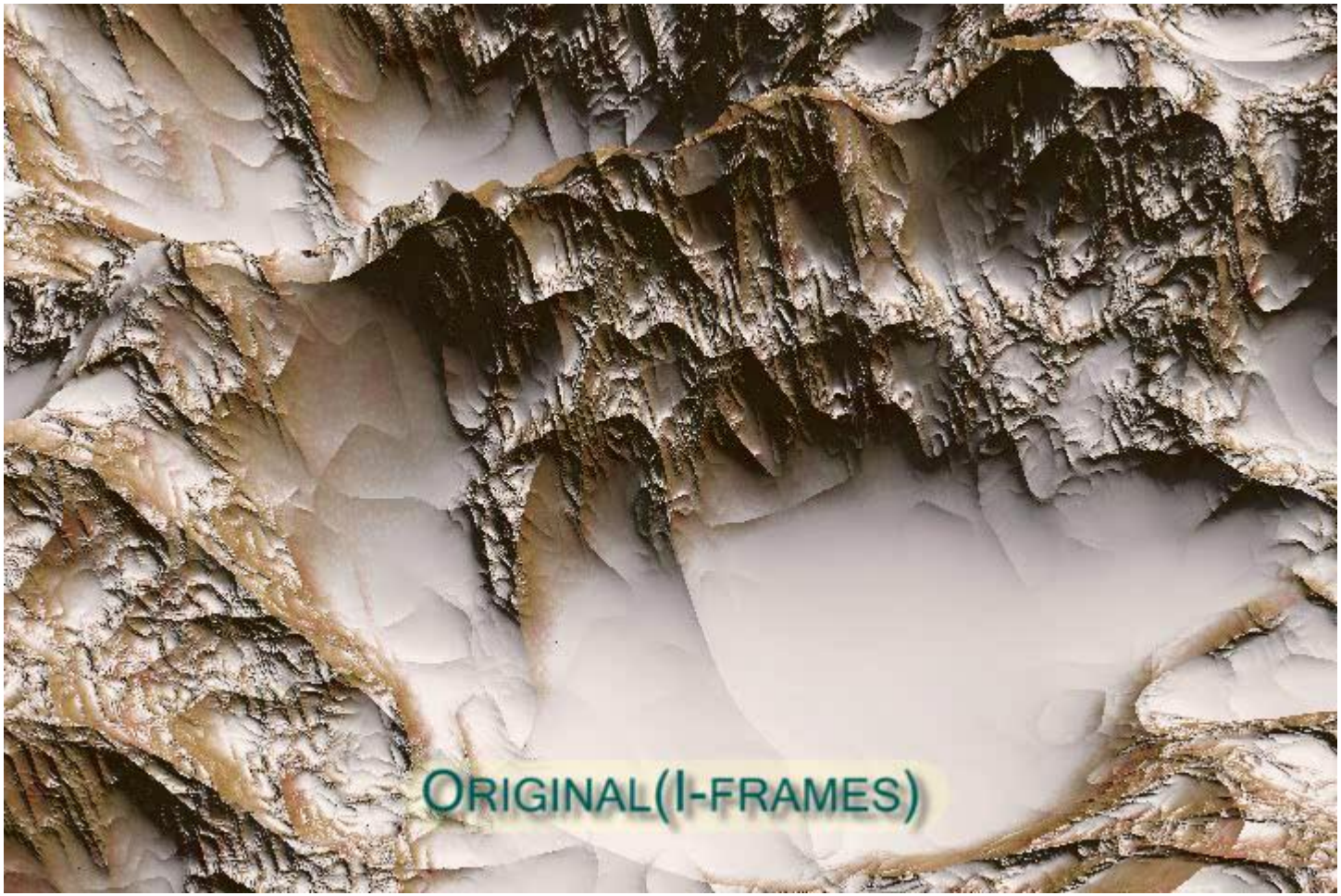
- Suitable scenarios:
  - Vertex-bound
  - Fill-bound scenes
  - Multi-pass rendering
  - Motion blur rendering
- Three B-frames per I-frame time step
- Image-based Bireproj:
  - 2-3ms for a B-frame
  - Pixel success rate: ≥99.6%

Results – the *walking* scene

- Fill-bound with an expensive noise shader
- Speed vs. reference: 2.9x (scene), 2.6x (image)

Results – the *terrain* scene

- Geometry bound (1M triangles)
- Speed vs. reference: 1x (scene), 2.8x (image)



Results – the *head* scene

- Multi-pass skin rendering [d'Eon and Luebke 2007]
- Speed vs. reference: 3.4x (scene), 2.9x (image)

Results – motion blur

- Accumulate 10 B-frames per I-frame
- Speed vs. reference: 5.4x (scene), 6.2x (image)

Improved shading interpolation

- Compared to uni-directional reprojection: Reduced popping artifacts with dynamic lighting and shadows

Conclusion

- General purpose rendering acceleration
- Real-time temporal upsampling
  - Bidirectional reprojection
  - Image-based iterative reprojection
- Advantages:
  - No need to reshade for disocclusion
  - Compatible with multi-pass and deferred rendering
  - Better dynamic shading interpolation
  - Effect of lag is small or negligible

Thanks!

- Acknowledgement
  - Piotr Didyk (for models and data)
  - NVIDIA and XYZRGB (for the human head assets)
  - NSF CAREER Award CCF-0747220
  - HK RGC GRF grants #619008 and #619509
  - INST grant from FAPERJ