
Realistic Image Synthesis

- Spatio-temporal Sampling and Reconstruction. Exploiting Temporal Coherence. Motion Blur. -

Philipp Slusallek
Karol Myszkowski
Gurprit Singh

Overview

- **Today**

- Sampling and reconstruction of spatio-temporal functions
- Motion compensation techniques
- Antialiasing techniques in animation
 - the amount of blur should be minimized
- Exploiting temporal coherence in global illumination
- Motion blur techniques
 - blur is intentionally introduced to model controllable shutter speed

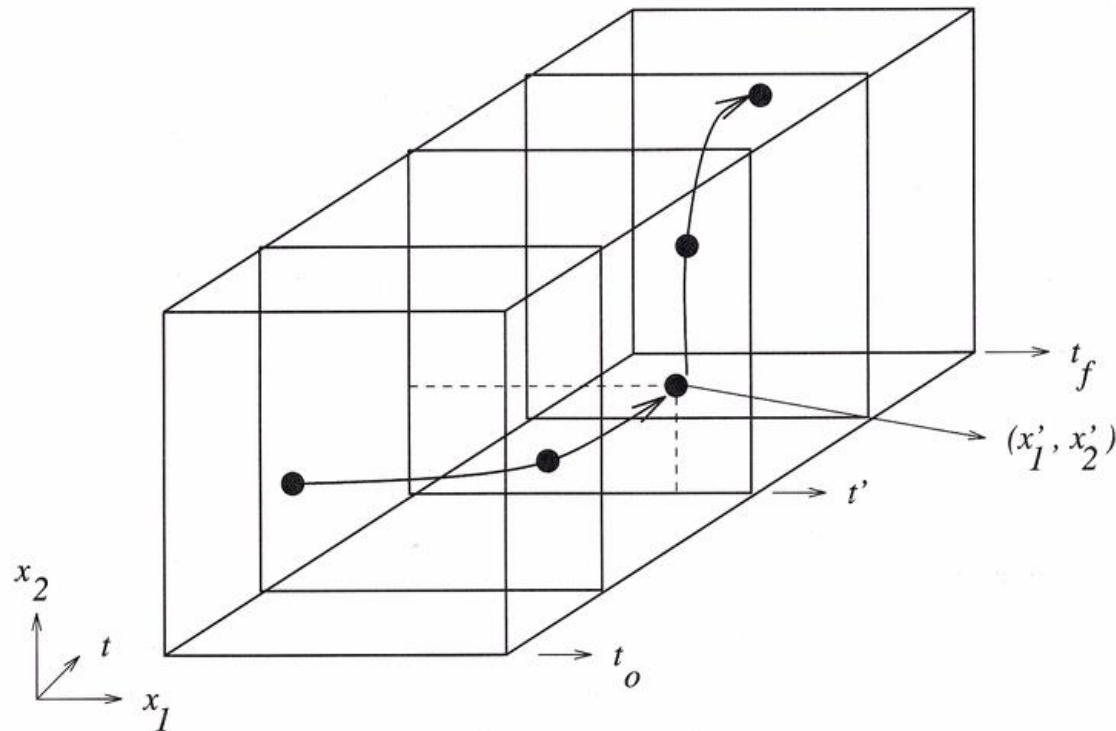
Reading Materials

Basic:

- **M. Tekalp, Digital Video Processing, Prentice Hall, Signal Processing Series, 1995**
- **K. Sung, A. Pearce, C. Wang, Spatial-Temporal Antialiasing, IEEE Transactions on Visualization and Computer Graphics, Vol.8, No.2, pp. 144-153, 2002**
- **M. Shinya, Spatial Antialiasing for Animation Sequences with Spatio-temporal Filtering, In Proceedings of ACM Siggraph 93, pp. 289-296**

Spatio-Temporal Fourier Spectrum

- **Assumption: the temporal variations in the image intensity pattern can be approximated by a simple motion model**



- **Motion trajectory - a curve in the 3D space (x_1, x_2, t) which is followed by a given point in image plane**

Spatio-Temporal Fourier Spectrum

- For simplicity let us consider the frame-to-frame intensity variations $s_c(x_1, x_2, t)$ only for the case of global motion with constant velocity (v_1, v_2) :

$$s_c(x_1, x_2, t) = s_c(x_1 - v_1 t, x_2 - v_2 t, 0) = s_0(x_1 - v_1 t, x_2 - v_2 t)$$

where $s_0(x_1, x_2)$ denotes the reference frame at $t_0 = 0$.

Fourier transform of spatio-temporal function $s_c(x_1, x_2, t)$:

$$S_c(F_1, F_2, F_t) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} s_c(x_1, x_2, t) e^{-j2\pi(F_1 x_1 + F_2 x_2 + F_t t)} dx_1 dx_2 dt =$$

$$\int_{-\infty}^{\infty} \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} s_0(x_1 - v_1 t, x_2 - v_2 t) e^{-j2\pi(F_1 x_1 + F_2 x_2 + F_t t)} dx_1 dx_2 dt =$$

$$\int_{-\infty}^{\infty} \int_{-\infty}^{\infty} s_0(x_1, x_2) e^{-j2\pi(F_1 x_1 + F_2 x_2)} dx_1 dx_2 \cdot \int_{-\infty}^{\infty} e^{-j2\pi(F_1 v_1 + F_2 v_2 + F_t) t} dt$$

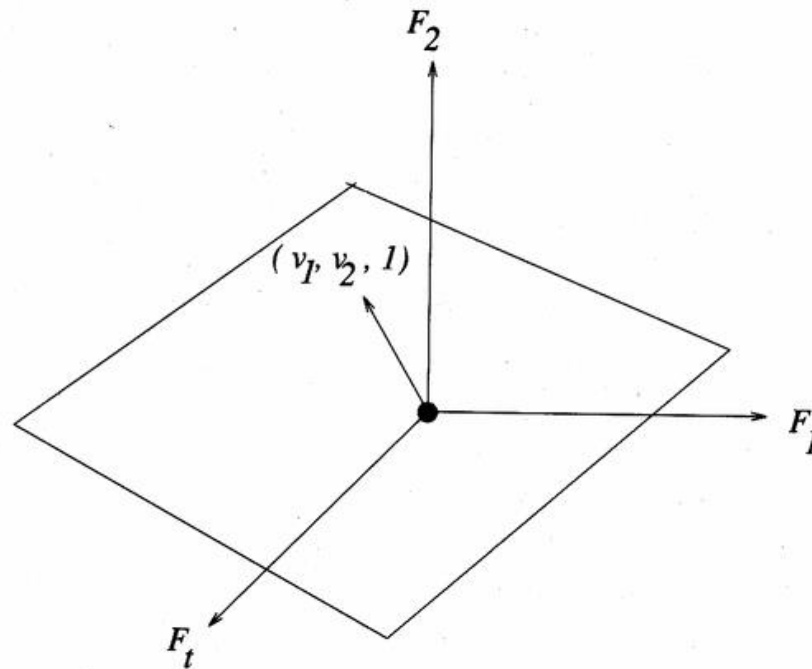
$$S_c(F_1, F_2, F_t) = S_0(F_1, F_2) \cdot \delta(F_1 v_1 + F_2 v_2 + F_t)$$

Spatio-Temporal Fourier Spectrum

$$S_c(F_1, F_2, F_t) = S_0(F_1, F_2) \cdot \delta(F_1 v_1 + F_2 v_2 + F_t)$$

thus the support of $S_c(F_1, F_2, F_t)$ is limited to a plane

$$F_1 v_1 + F_2 v_2 + F_t = 0$$



The spectral support of animation function $s_c(x_1, x_2, t)$ with global, uniform velocity motion

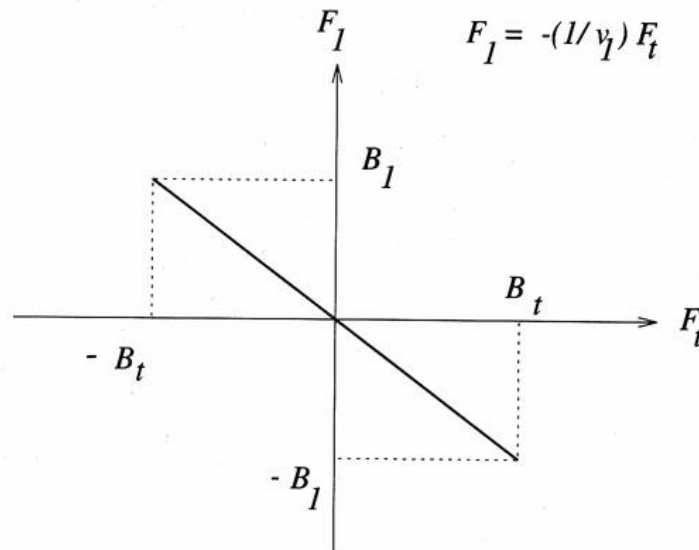
Spatio-Temporal Fourier Spectrum

For the bandlimited image $s_0(x_1, x_2)$ such that

$$S_0(F_1, F_2) = 0 \text{ for } |F_1| > B_1 \text{ and } |F_2| > B_2$$

the animation sequence $s_c(x_1, x_2, t)$ is also bandlimited in the temporal domain

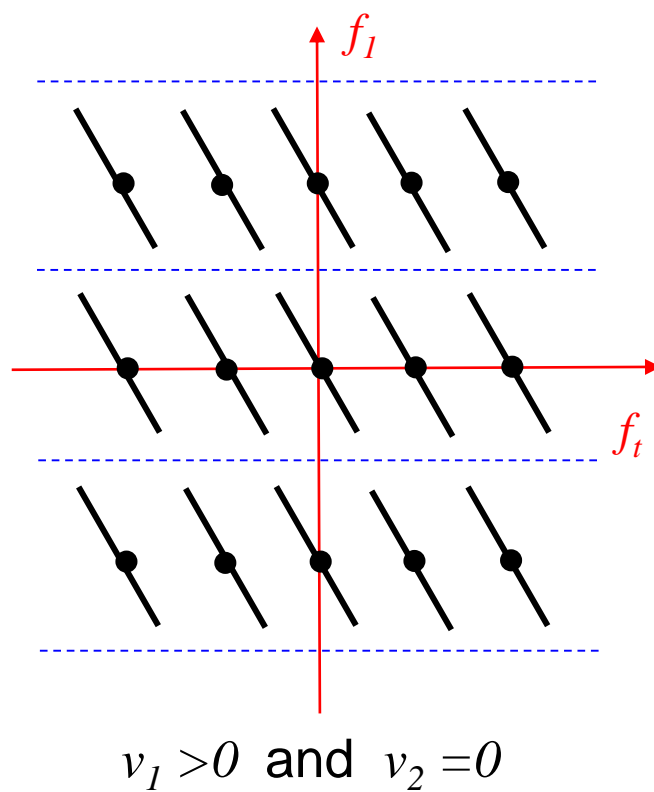
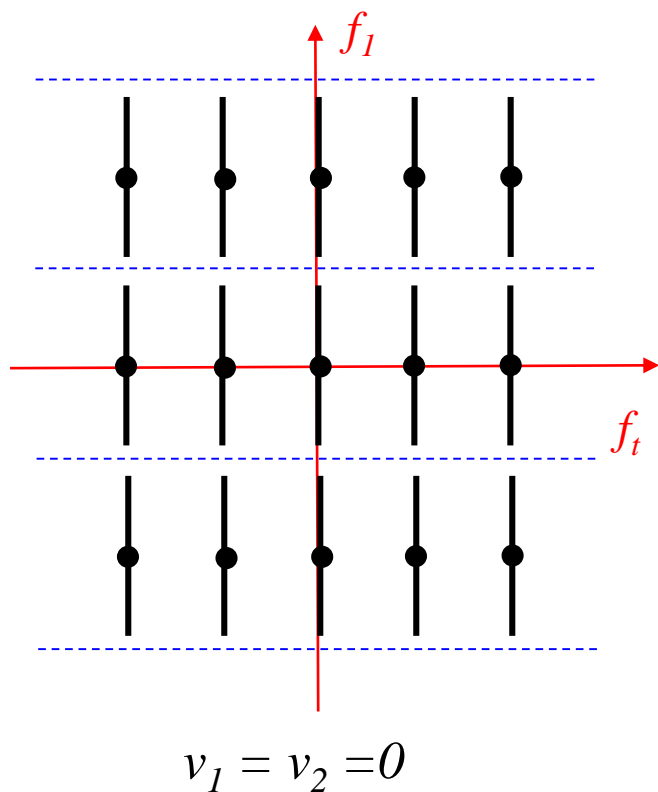
$$S_c(F_1, F_2, F_t) = 0 \text{ for } |F_t| > B_t \text{ where } B_t = B_1 v_1 + B_2 v_2$$



Projection of the $S_c(F_1, F_2, F_t)$ support into (F_1, F_t) plane for $F_1 v_1 + F_t = 0$

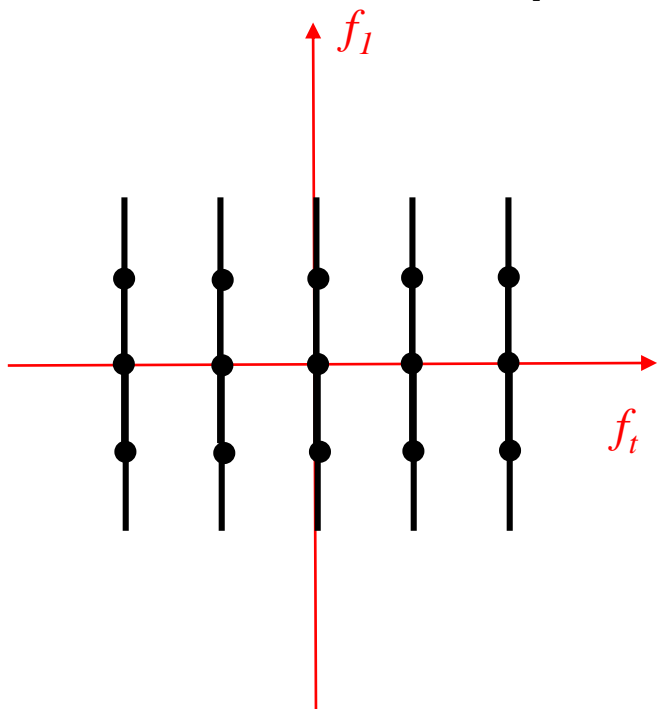
Sampling on a Spatio-Temporal Lattice

- Sampling frequencies above the Nyquist limit, ie. above $2B_1$, $2B_2$, and $2B_t$

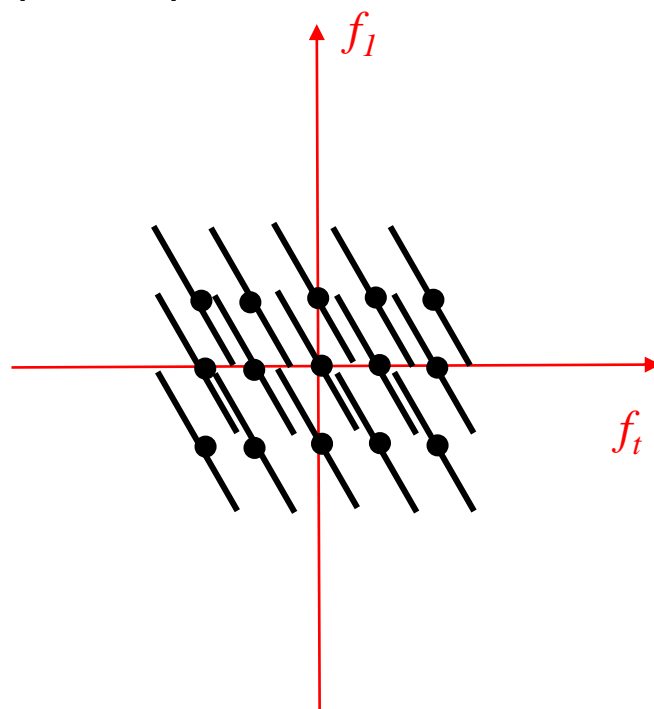


Sampling on a Spatio-Temporal Lattice

- **Sampling frequencies under the Nyquist limit**
 - $v_1 = v_2 = 0$: spectral replicas overlap
 - aliasing-free reconstruction of $s_c(x_1, x_2, t)$ is not possible
 - $v_1 > 0$ and $v_2 = 0$: spectral replicas interleaved
 - **ideal reconstruction possible!** (requires special reconstruction filter)



$$v_1 = v_2 = 0$$

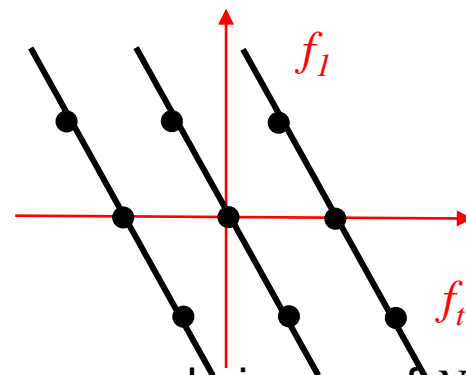


$$v_1 > 0 \text{ and } v_2 = 0$$

Sampling on a Spatio-Temporal Lattice

- **Critical velocities:**

- Frequency domain interpretation:
 - velocities that for a given spatial bandwidth of $s_c(x_1, x_2, t)$ and sampling lattice result in overlapping of the replicas.
- Spatio-temporal domain interpretation
 - motion trajectory cannot pass through an existing sample in any of N consecutive frames used for the reconstruction (otherwise such a sample does not provide „new“ information)
 - Example:
 - Assumption: frames are spatially sub-Nyquist sampled by a factor of 2.
 - Samples collected for four consecutive frames make possible ideal reconstruction of frame k under the condition that any motion trajectory does not pass through an existing sample for the next three frames $k+1$, $k+2$, and $k+3$.
 - Velocities $v=0$, $v=1$, $v=1/2$, and $v=1/3$ (measured in pixels per frame) are the critical velocities in this example



Motion-compensated Filtering

- **Optimum filtering strategy for noise removal and reconstruction**
 - samples along motion trajectory contain different noisy realizations of the reconstructed value
- **Algorithm: Filtration the input frame at point (x_1, x_2, t)**
 - 1-D signal along the motion trajectory traversing (x_1, x_2, t) is convolved with 1-D filter function operating along this trajectory within the support of K frames
 - motion-compensated low-pass filter with the cutoff frequencies B_1 , B_2 , and B_t

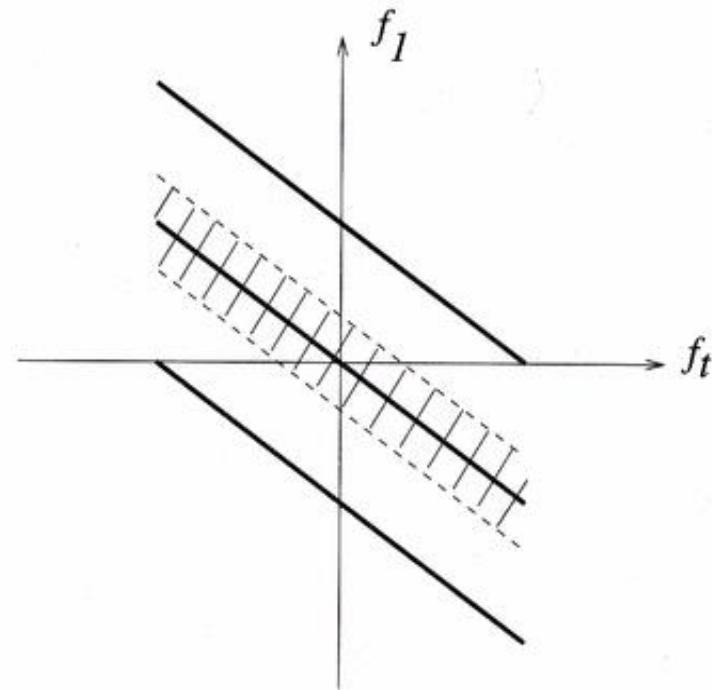
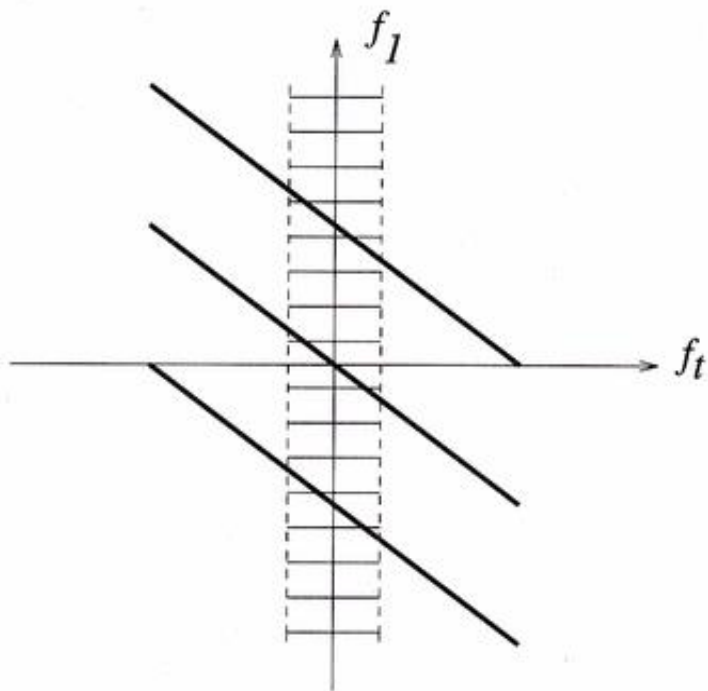
$$H(F_1, F_2, F_t) = \begin{cases} 1 & F_1 < |B_1|, F_2 < |B_2|, \text{ and } -v_1 F - v_2 F_2 - B_t < F_t < -v_1 F - v_2 F_2 + B_t \\ 0 & \text{otherwise} \end{cases}$$

the filter spatio-temporal frequency support matches the support of the reconstructed animation function which is limited to a plane

$$F_1 v_1 + F_2 v_2 + F_t = 0$$

Motion-compensated Filtering

- Filtering without motion compensation leads to blurry images for quickly moving objects or camera



Motion-compensated Filtering

- Adaptive filtering algorithm:
 - **Assumption: Intensity of samples along a precisely computed motion trajectory should be similar**
 - **This assumption may not hold because of**
 - inaccuracies in the computation of motion trajectory
 - occlusions/disocclusions (lead to motion trajectory bifurcations)
 - view-dependent changes in shading for glossy and specular objects
 - changes in the scene, eg., lighting, object deformations
 - **If this assumption does not hold then**
 - **stop collecting samples in the temporal domain and normalize the reconstruction filter weights due to its narrower support**
 - result: increase of noise
 - **or instead of collecting samples in the temporal domain, collect samples of similar intensity in the spatial domain and process them using the reconstruction filter**
 - result: increase of blur

Motion-compensated Filtering

- Occlusion problem

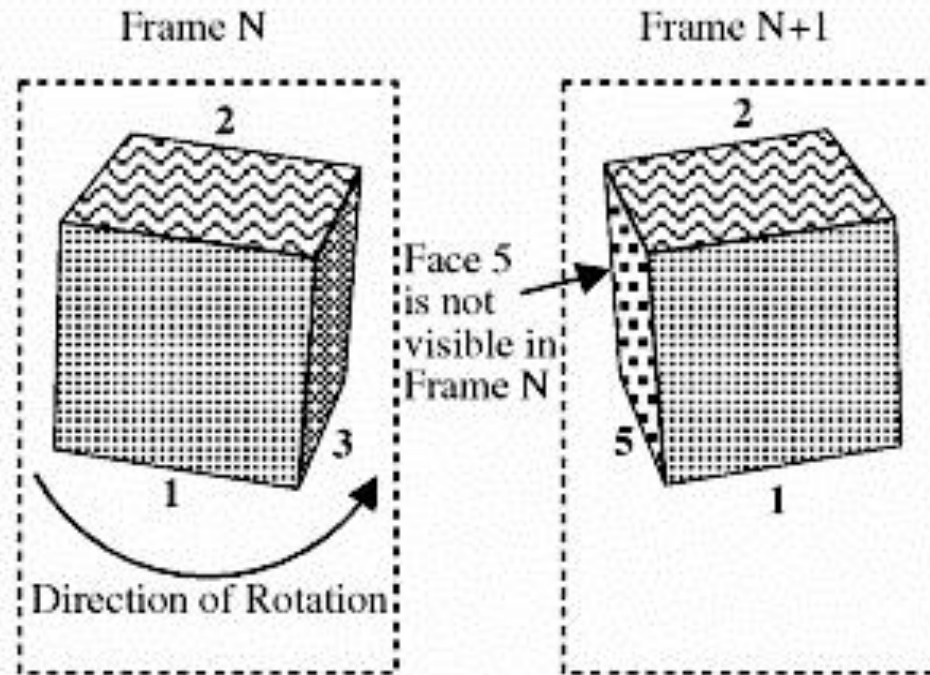


Figure 2: While face 5 of this object is visible in frame $N + 1$, it is occluded by faces 1 and 2 in frame N . Back-projecting pixels that fall in face 5 will yield invalid optical flow vectors.

Motion-compensated Filtering

- All derivations so far were performed under the assumption of constant velocity for the whole image plane
 - **in practice these derivations remain valid for any coherent block of pixels when applied over a short period of time**
- The critical issue is the accuracy of motion trajectories
 - **for natural image sequences difficult to acquire**
 - optical flow computation
 - **for synthetic images easy to compute even with subpixel accuracy**
 - camera motion compensation
 - object motion compensation

Natural Image Sequences

- **The optical flow derivation**

- mathematical model: „intensity does not change along motion trajectory“

$$\frac{ds_c(x_1, x_2, t)}{dt} = 0$$

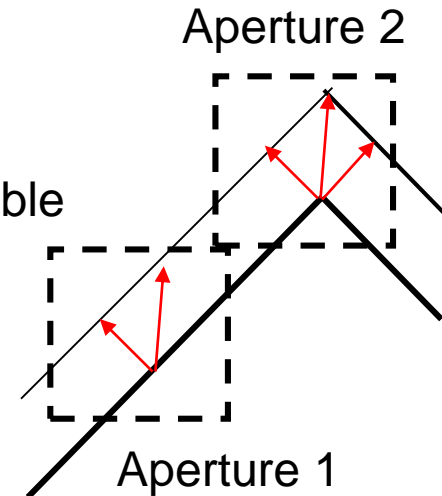
and after applying the chain differentiation rule

$$\frac{\partial s_c(x_1, x_2, t)}{\partial x_1} v_1(x_1, x_2, t) + \frac{\partial s_c(x_1, x_2, t)}{\partial x_2} v_2(x_1, x_2, t) + \frac{\partial s_c(x_1, x_2, t)}{\partial t} = 0$$

where $v_1(x_1, x_2, t) = dx_1 / dt$ and $v_2(x_1, x_2, t) = dx_2 / dt$

- **Problems**

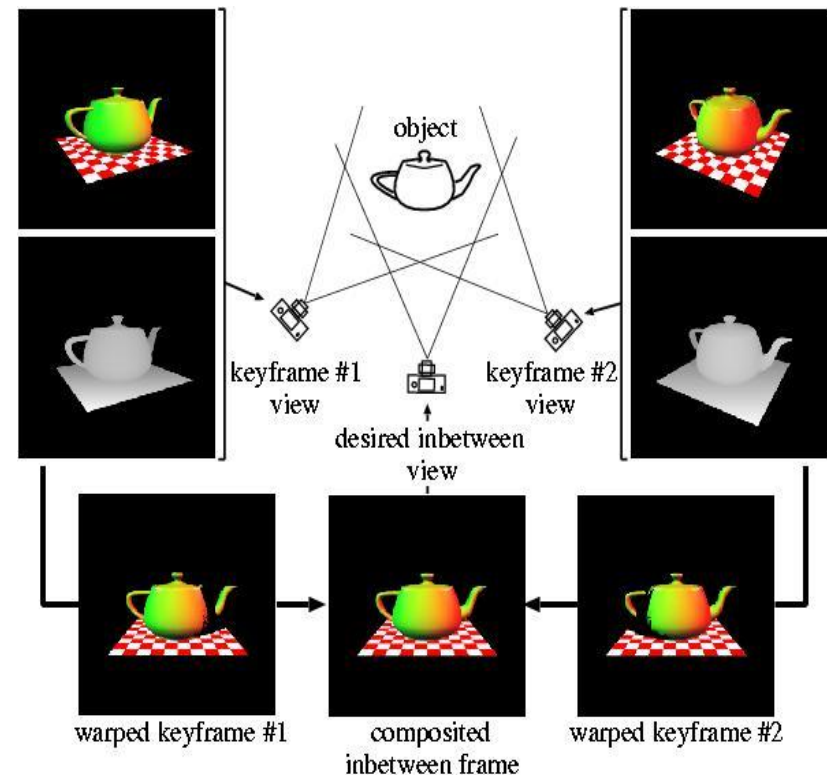
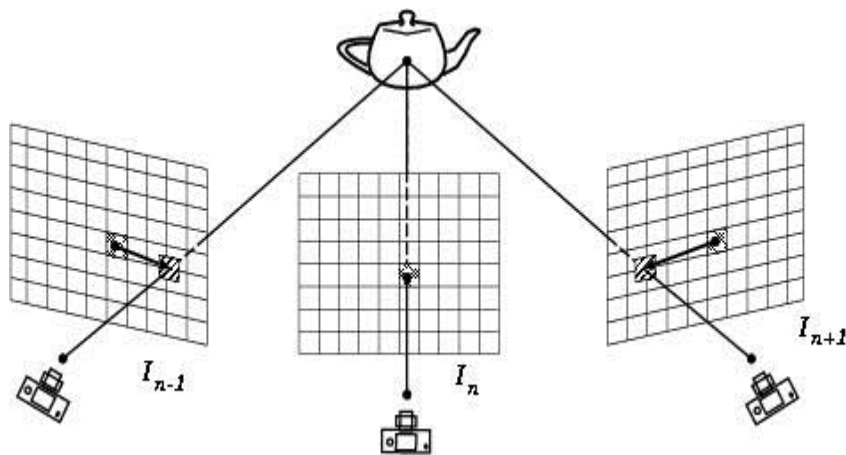
- image gradients are required
 - surfaces with textures or complex shading are desirable
- changes in shading can be confusing
- lack of coherence due to occlusions/dissocclusions
- aperture problem



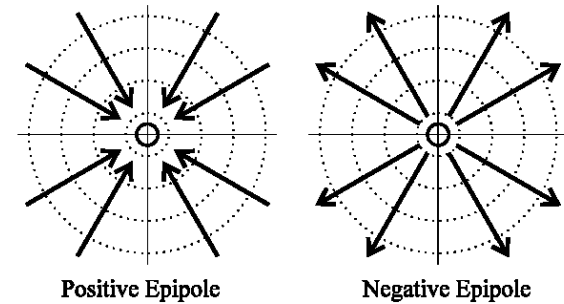
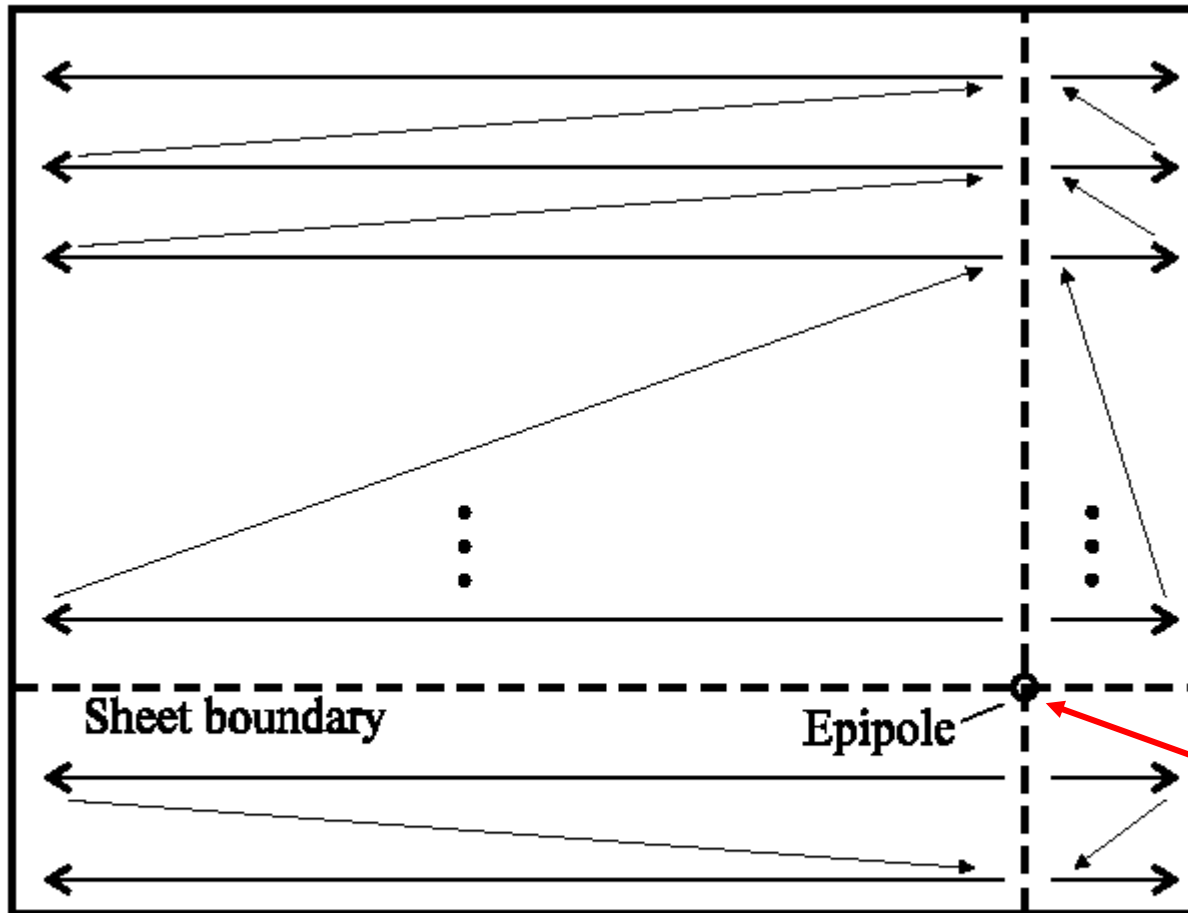
Synthetic Image Sequences

- **Camera motion compensation using Image-Based Rendering techniques**

- Can be performed in backward and forward directions
- Efficiency: McMillan's occlusion coherent ordering algorithm
 - depth comparisons not required
- Required input data
 - Camera parameters
 - Depth data for every pixel
- Very precise



McMillan's Occlusion Coherent Ordering



The scanning order as for the **negative epipole** – the viewer moves away from the reference view position and the rendered scene

The projection of the output image view position onto the reference-image plane

The reference image can be divided into four occlusion-compatible sheets. Each sheet is traversed in a raster-like order.

Synthetic Image Sequences

- **Rigid object motion compensation**
 - Required input data
 - Camera parameters
 - Depth data for every pixel
 - Transformations describing rigid object motion
 - Very precise

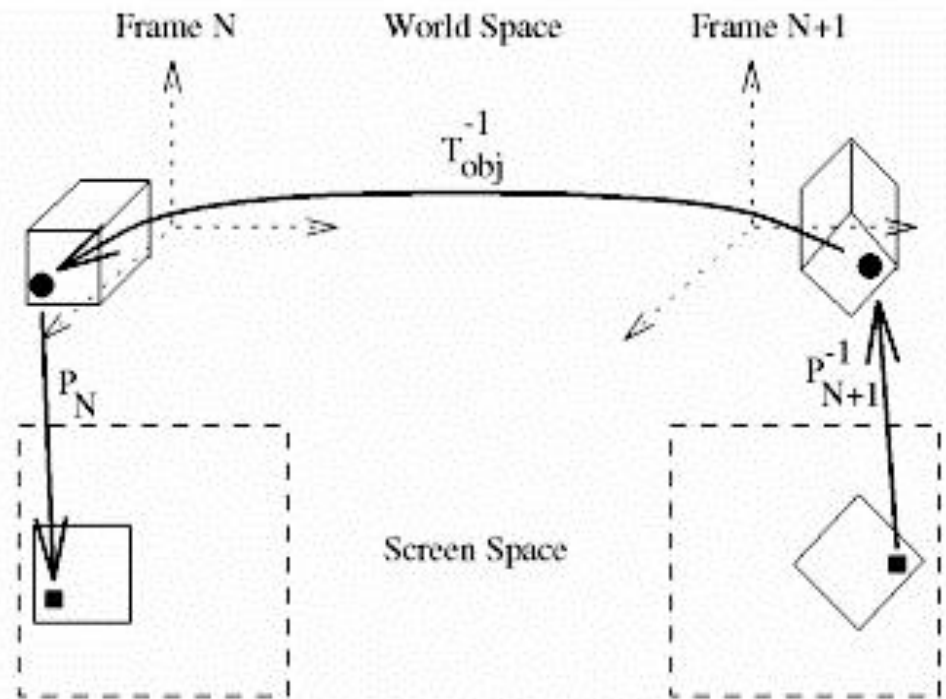


Figure 1: Backprojecting a pixel from frame $N + 1$ into frame N is done by multiplying the frame $N + 1$ pixel position through three matrixes. The inverse projection matrix P_{N+1}^{-1} transforms the pixel into frame $N + 1$ world space. The matrix T_{obj}^{-1} transforms it back to frame N world space and the projection matrix P_N transforms it into the frame N screen space.

$$q^N = P_N T_{obj}^{-1} P_{N+1}^{-1} q^{N+1}$$

Synthetic Image Sequences

- **Handling critical velocities (in particular $v=0$)**
 - jittered sampling in the image plane eliminates correlations between object motion (including periodic motions) and sampling
- **Handling dynamic lighting (including moving shadows and highlights)**
 - find motion compensated surface samples with computed textures but without shading computation (deferred lighting computation)
 - perform shading computation for all those samples for a given frame (time t)
 - ⊗ this may lead to significant cost increase due to repeating lighting computation for multiple samples per pixel for each frame
- **Handling non-rigid objects (parametric deformations are assumed)**
 - store surface parameter value s for each pixel in frame k
 - find surface points corresponding to s for neighboring frames
 - perform shading computation for all those points for frame k

Motivation: Temporal Coherence

GI algorithms designed for dynamic environments

- Exploiting temporal coherence (reuse information from previous frames):
 - + Avoid redundant computation
 - + Reduce temporal aliasing
 - Lack of generality
 - May require different processing depending on interactive changes in the scene
- Brute force (compute each frame from scratch):
 - + Very general
 - Can immediately handle all types of scene changes
 - Global illumination is costly so many processors might be required
 - Temporal aliasing might be an issue

Interactive Requirements

- **Everything may change in the scene**
 - Geometry, lighting, material properties, camera parameters, ...
 - Changes are not known a priori

- **Aiming at fast feedback to user change:**
 - A minimum frame rate should be ensured
 - Some applications may require constant frame rate
 - Image quality/precision can be traded for faster response time
 - Sudden/unexpected changes distracting the user should be avoided, e.g.,
 - popping,
 - changing frame rate,
 - reducing image quality.

Exploiting Temporal Coherence

- **Different coherence levels can be considered**
 - Reusing complete global illumination samples
 - Image space (pixels): **Render Cache algorithm**
 - Reusing photons
 - Reusing visibility information
 - Reusing seeds of random generator
 - All photon paths are computed from scratch
 - Seeds should be stored for each photon paths
 - Does not work well for paths originating at the eye position when camera parameters are changing

Exploiting Temporal Coherence

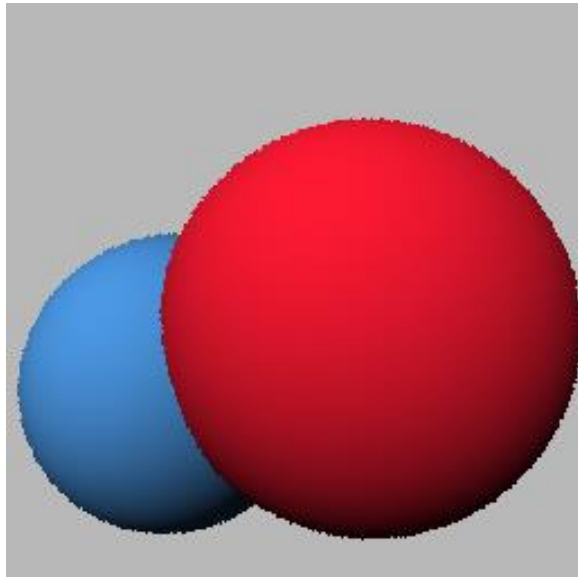
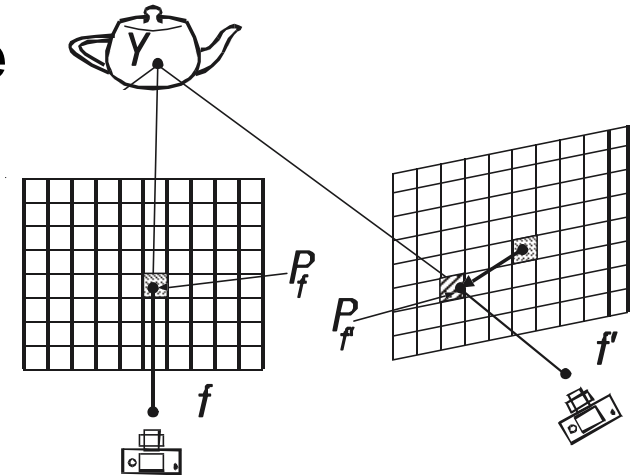
- **The higher level at which the coherence is exploited the more computational savings can be expected, but chances of reusing information may get lower, e.g.:**
 - Samples for glossy surfaces cannot be re-used for moving camera but a photon path contribution can be potentially reweighted by the current BRDF value
 - When a light source is changed then only those photon paths that are linked to this light source need to be updated
 - and so on ...
- **Using the coherence at the lower levels requires specialized data structures and functions to handle each type of changes in the scene.**

Render-Cache : Principles [Walter'99]

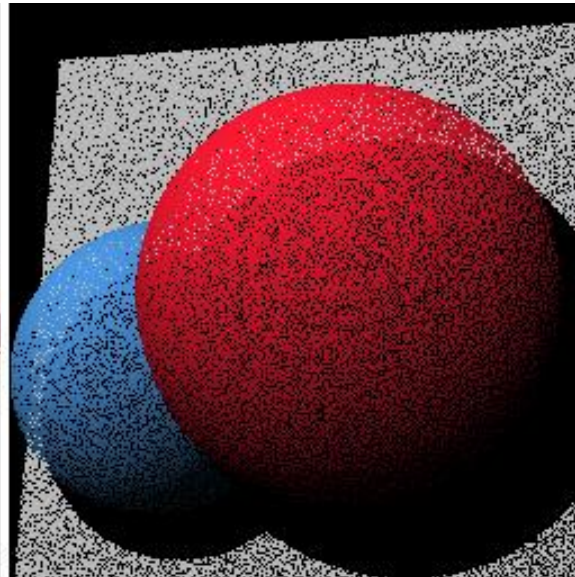
- **Decouples global illumination computations from image rendering (frameless rendering)**
- **Reconstructs the illumination from a sparse set of samples in image space**
- **Purely software approach based on point reprojection and adaptive sampling**
- **Each sample point is stored with additional information:**
 - 3D position (located on surfaces)
 - Color
 - Age
 - Object identifier

Render Cache: Reprojection

- **Reproject points into current frame**
 - Occlusion errors
 - Holes in data



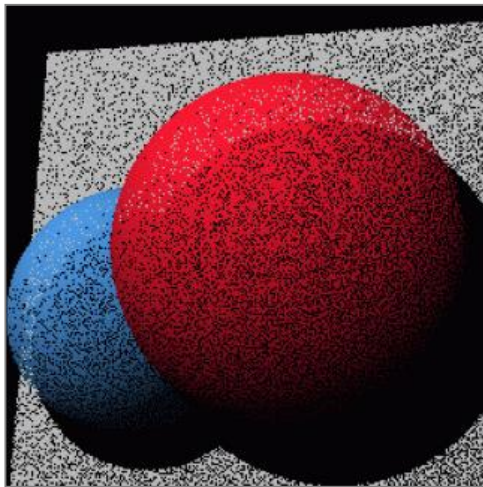
Initial view



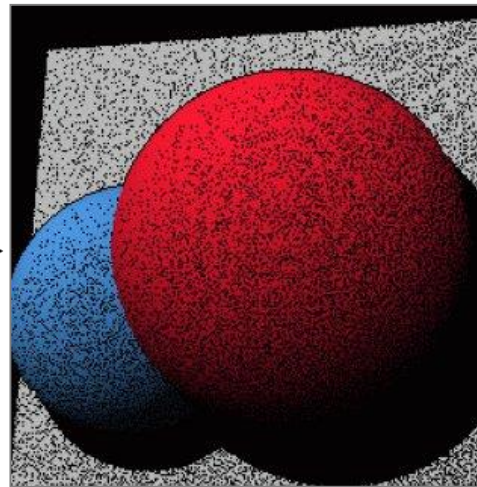
After reprojection

Render Cache: Image Reconstruction

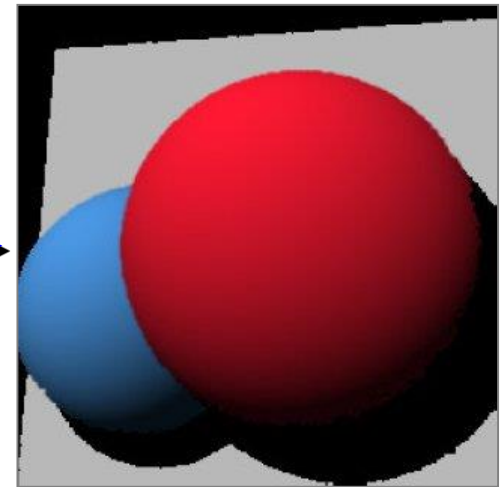
- After reprojection, occluded points are removed by a depth-culling heuristic
- Holes filled by interpolation and filtering
 - Fixed size kernels, 3x3 and 7x7



Reprojection



Occlusion cull

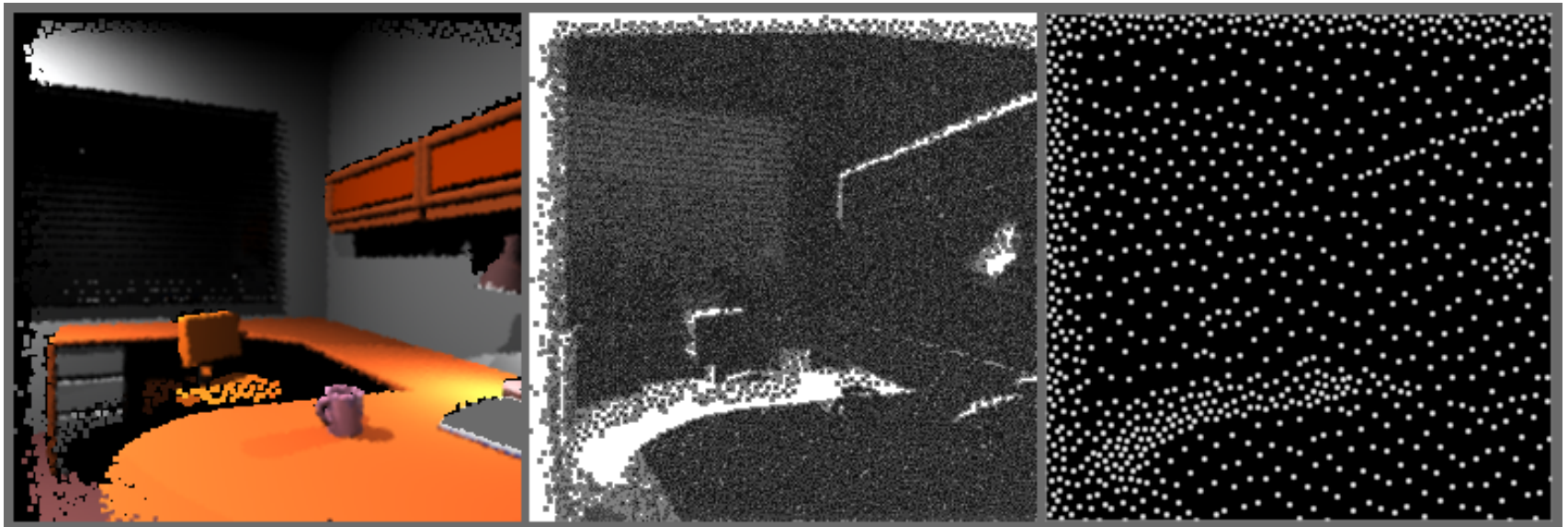


Interpolation

Render Cache: Sample Update

- **Priority image for sampling**
 - High priority for sparse regions
 - High priority for old points
- **Convert priority image to sparse set of locations to be rendered**
 - Uses error-diffusion dither
- **Also uses predictive sampling**
 - Try to sample new regions just before they become visible

Render Cache: Sample Update



Displayed image

Priority image

Requested pixels

Render Cache: Sample Update

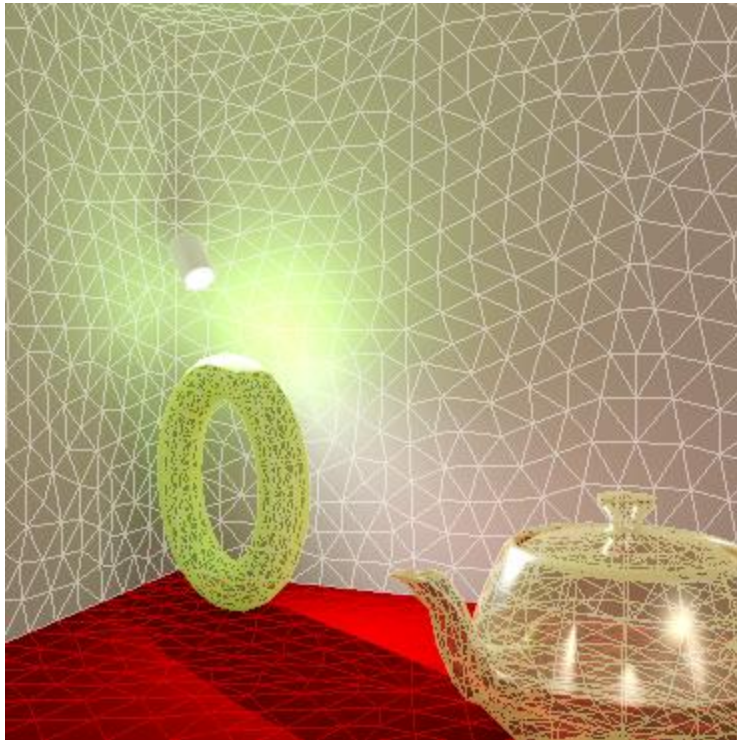
- **Recomputes old samples to detect changes**
 - Nearby points are aged to raise priority and cause point invalidation
- **Object motion**
 - Associated points can be transformed along with the object

Render Cache: Discussion

- **Designed for ray tracing but can also be used with path tracers**
- **Scalable with high image resolution if carefully implemented [Walter'02]**
- **Rapid movements may cause distracting artifacts**

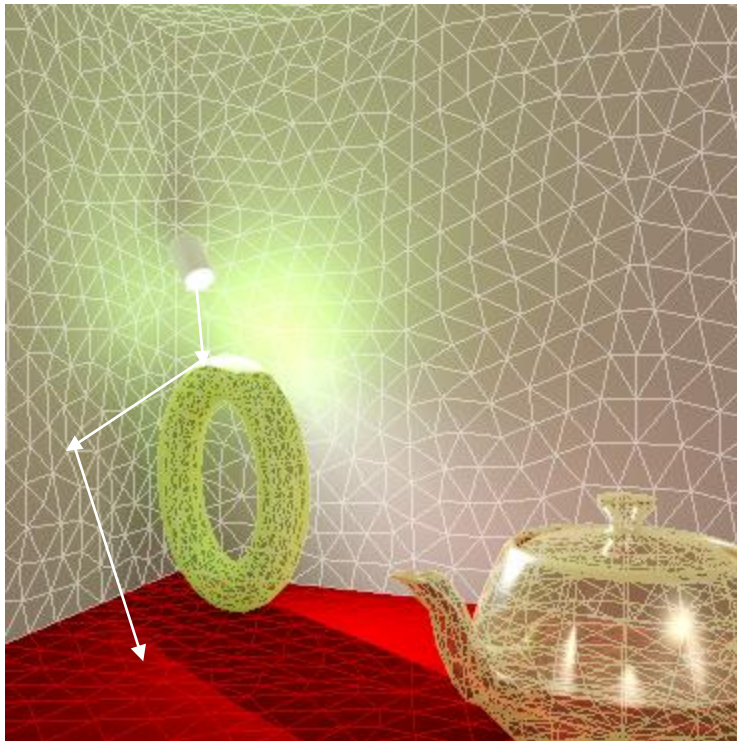
- **Downloadable version**
 - <http://www.graphics.cornell.edu/research/interactive/rendercache/>

Spatio-Temporal Density Estimation



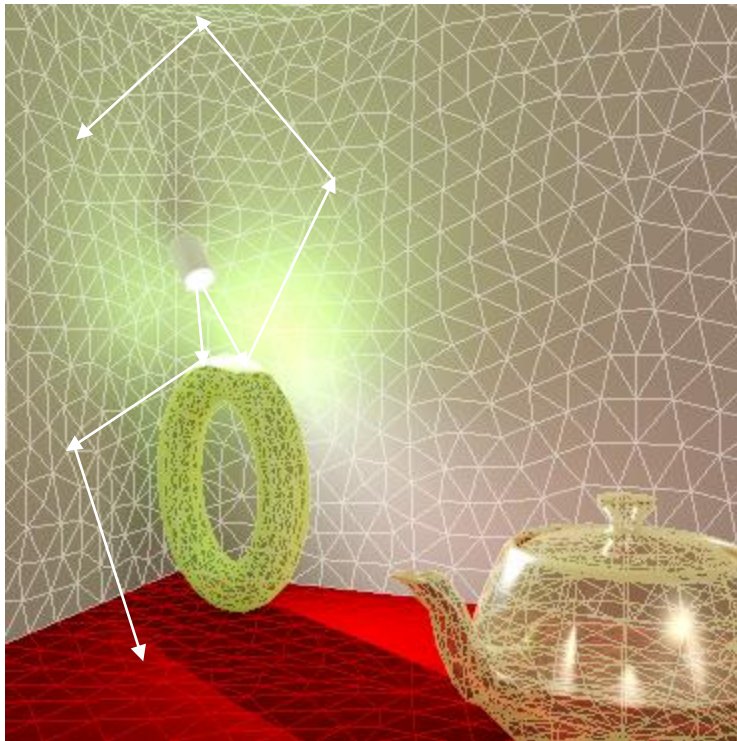
- **View-independent light source photon tracing**

Spatio-Temporal Density Estimation



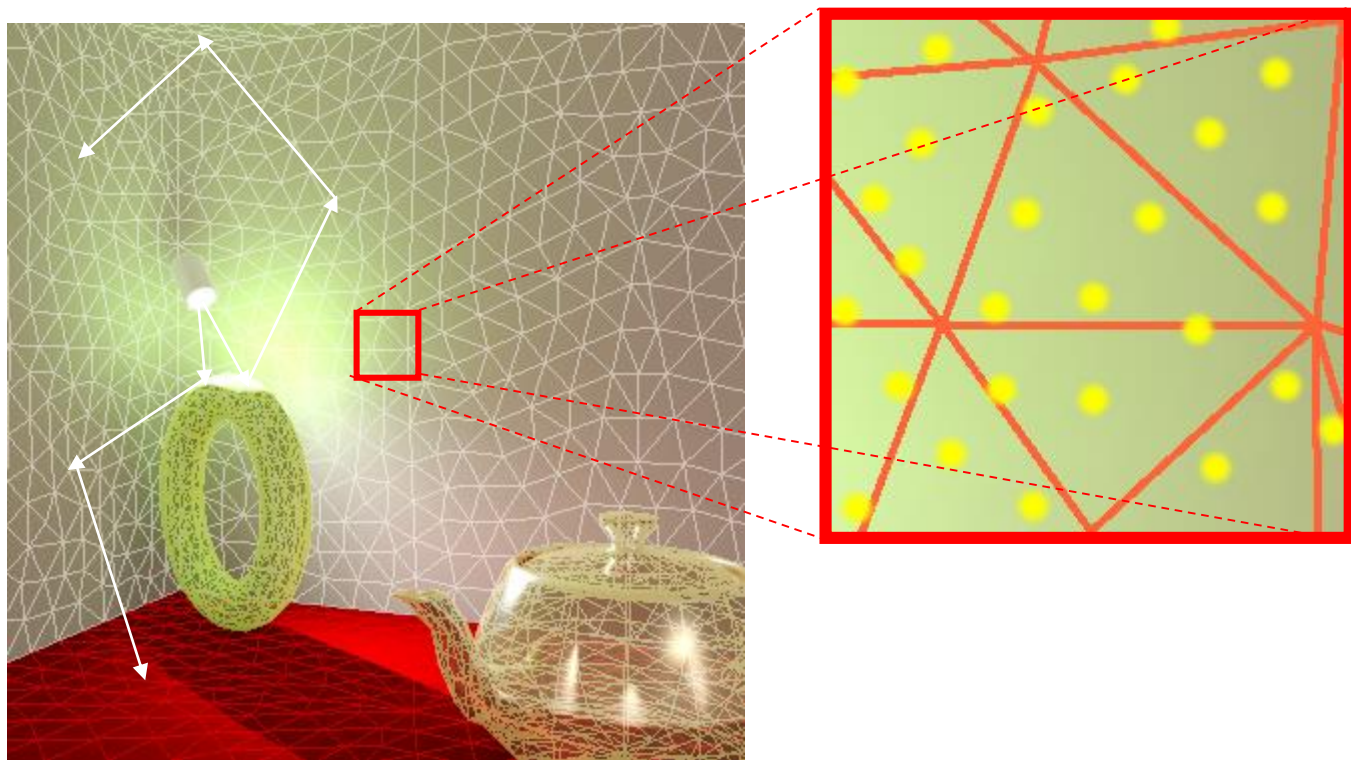
- **View-independent light source photon tracing**

Spatio-Temporal Density Estimation



- **View-independent light source photon tracing**

Spatio-Temporal Density Estimation

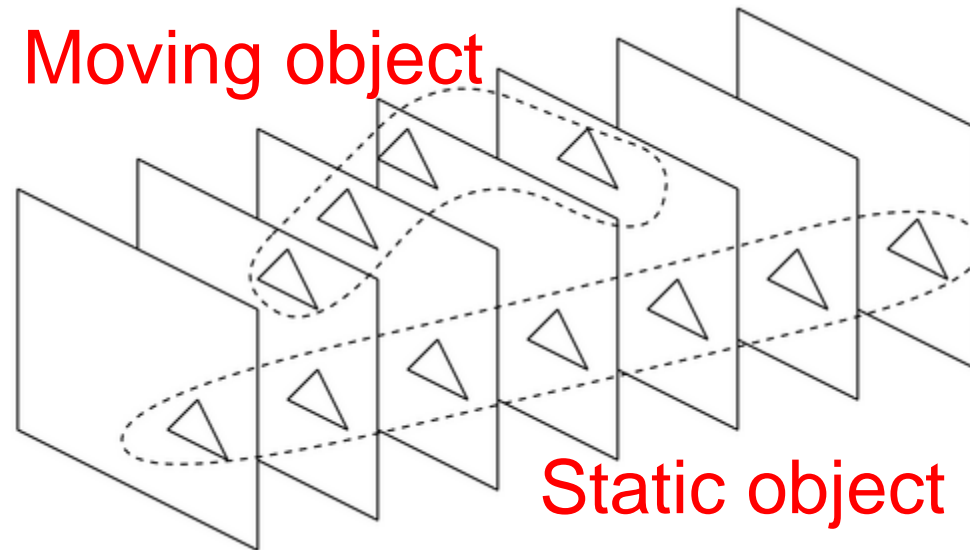


- **Extending photon processing into the temporal domain**

Temporal Photon Processing

- **Contradictory Requirements:**

- Maximize the number of photons collected in the temporal domain to reduce the stochastic noise.
- Minimize the time interval in which the photons were traced to avoid collecting invalid photons.



Temporal Photon Processing

- **Energy-based stochastic error metric**
 - Steers the photon collection in the temporal domain
 - Computed for each mesh element and for all frames
- **Perception-based animation quality metric**
 - Decides upon the stopping condition
 - Computed once per animation segment

Energy-based Error Metric

- **Problem:**

- *How to distinguish the actual changes in lighting from the stochastic noise?*

- **We assume that hitting a mesh element by photons can be modeled by the Poisson distribution.**

- For the mean number μ of hit points the standard deviation

$$\sigma = \sqrt{\mu}$$

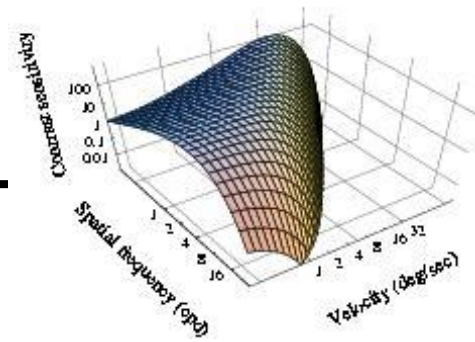
for $\mu = 0$ we assume $\sigma = 1$

- If the number of photons x hitting a mesh element does not satisfy the condition

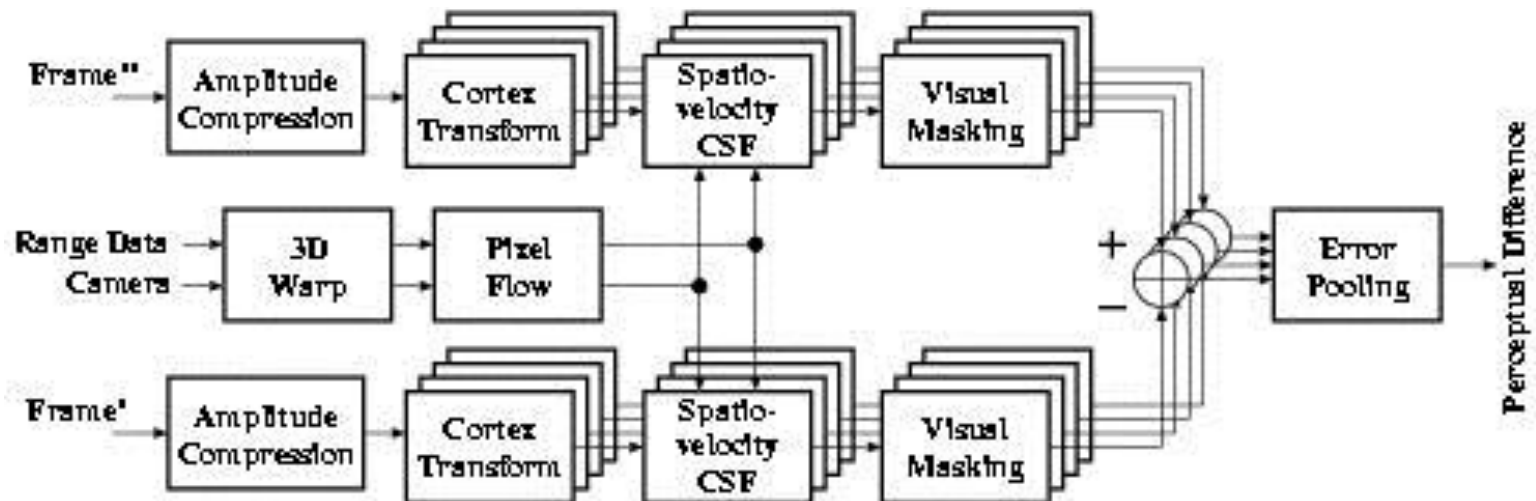
$$\mu - k\sigma \leq x \leq \mu + k\sigma$$

the photon collection for this mesh element is disabled (e.g., $k = 2$).

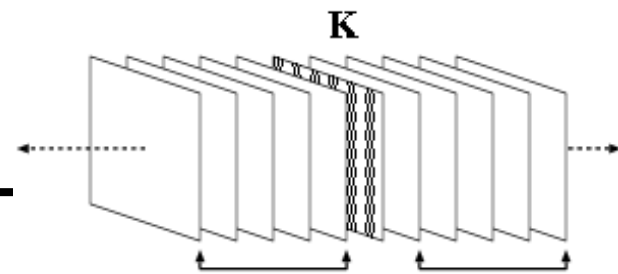
Animation Quality Metric (AQM)



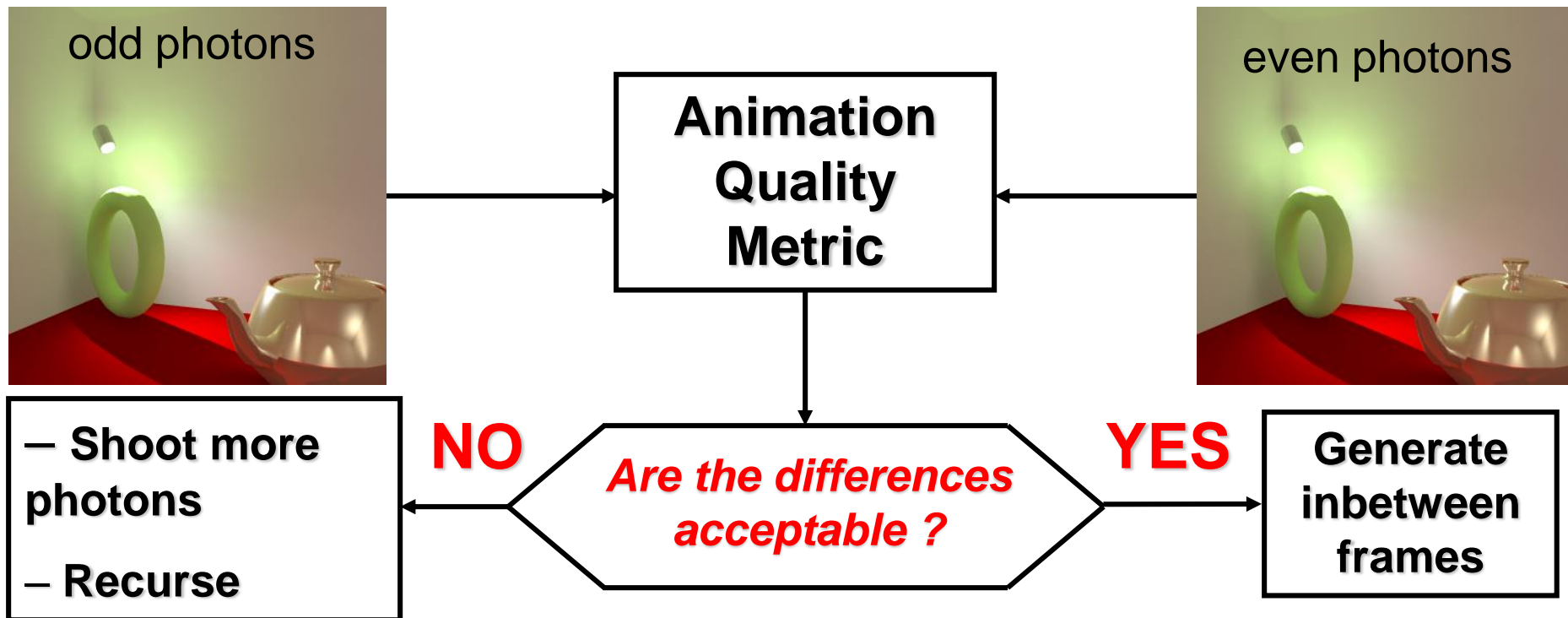
- Computes the map of visible differences between two input animation frames
- Human Vision System modeling:
 - Weber law
 - Spatio-velocity Contrast Sensitivity Function
 - Visual masking



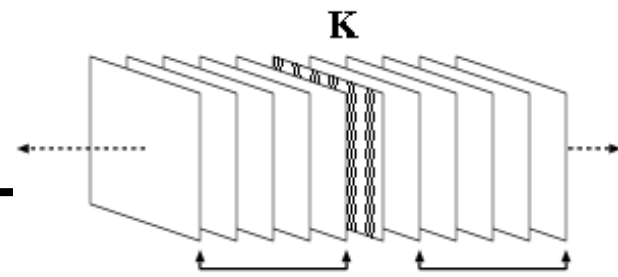
Perception-Based Guidance



- **AQM: Decides upon the computation stopping condition**
 - Computed once per animation segment for a central frame K



AQM Processing



1. **Select the central frame K for a given animation segment.**
2. **Split all photons collected in the temporal domain for this frame into two halves and compute two corresponding images.**
3. **Use the AQM to predict the perceivable differences between these two images.**
4. **If the noise is perceived for more pixels than a certain threshold value the number of photons is increased.**

Space-time Density Estimation Algorithm

- 1. Initialization: determine the initial number of photons per frame.**
- 2. Adjust the animation segment length depending on temporal variations of indirect lighting which are measured using energy-based criteria.**
- 3. Adjust the number of photons per frame based on the AQM response to limit the perceivable noise.**
- 4. Spatio-temporal reconstruction of indirect lighting.**
- 5. Spatial filtering step.**

Video: Scene Room

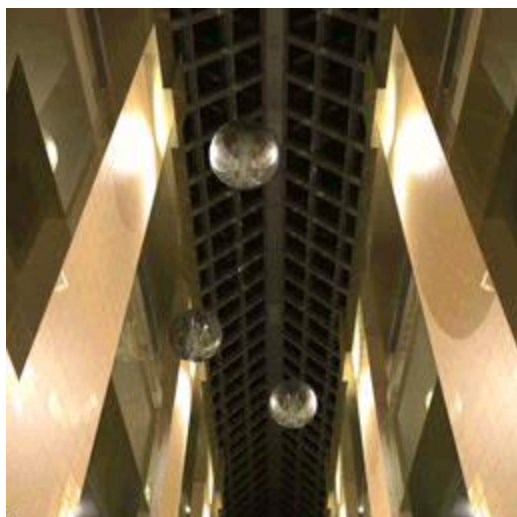


Temporal processing: Off



Temporal processing: On

Video: Scene Atrium



Temporal processing: Off



Temporal processing: On

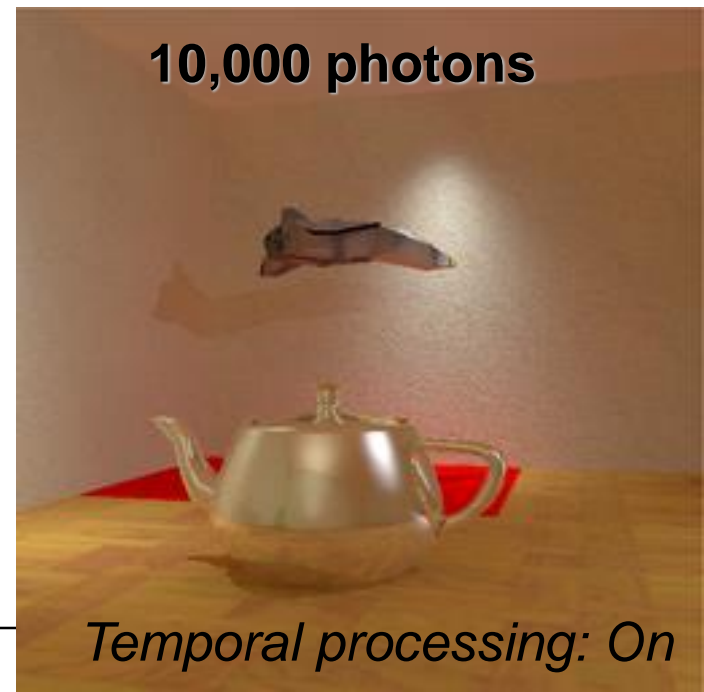
Discussion

- + Significantly reduced the number of photons to be shot per frame
- + Drastically reduced temporal aliasing
- Limited spatial resolution of mesh reconstructed lighting
- For quickly changing indirect lighting temporal processing can be limited
 - Spatial filtering can be performed at the expense of losing spatial lighting details
 - More photons can be shot at the expense of performance loss

25,000 photons



10,000 photons



Space-time Architecture: Principle

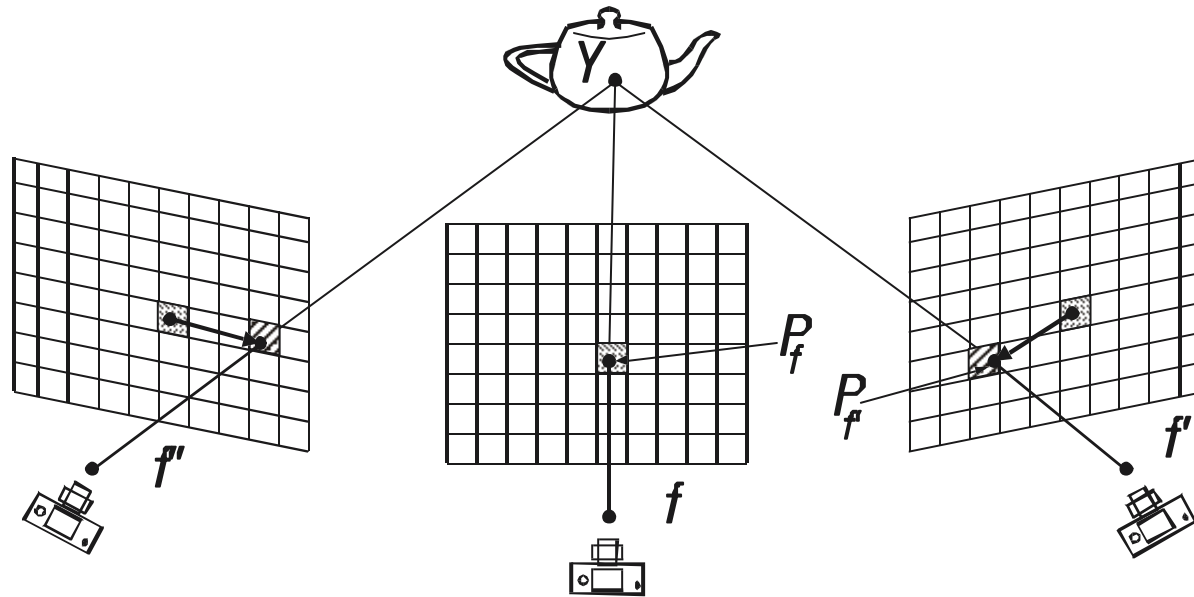
- **Compute samples using a variant of Path-Tracing**
- **Pixels color = mean of sample values**
- **2 types of samples:**
 - Native samples:
 - Expensive, computed from scratch
 - Recycled samples:
 - Cheap, based on previous computations (using reprojections)

- **Algorithm outline:**

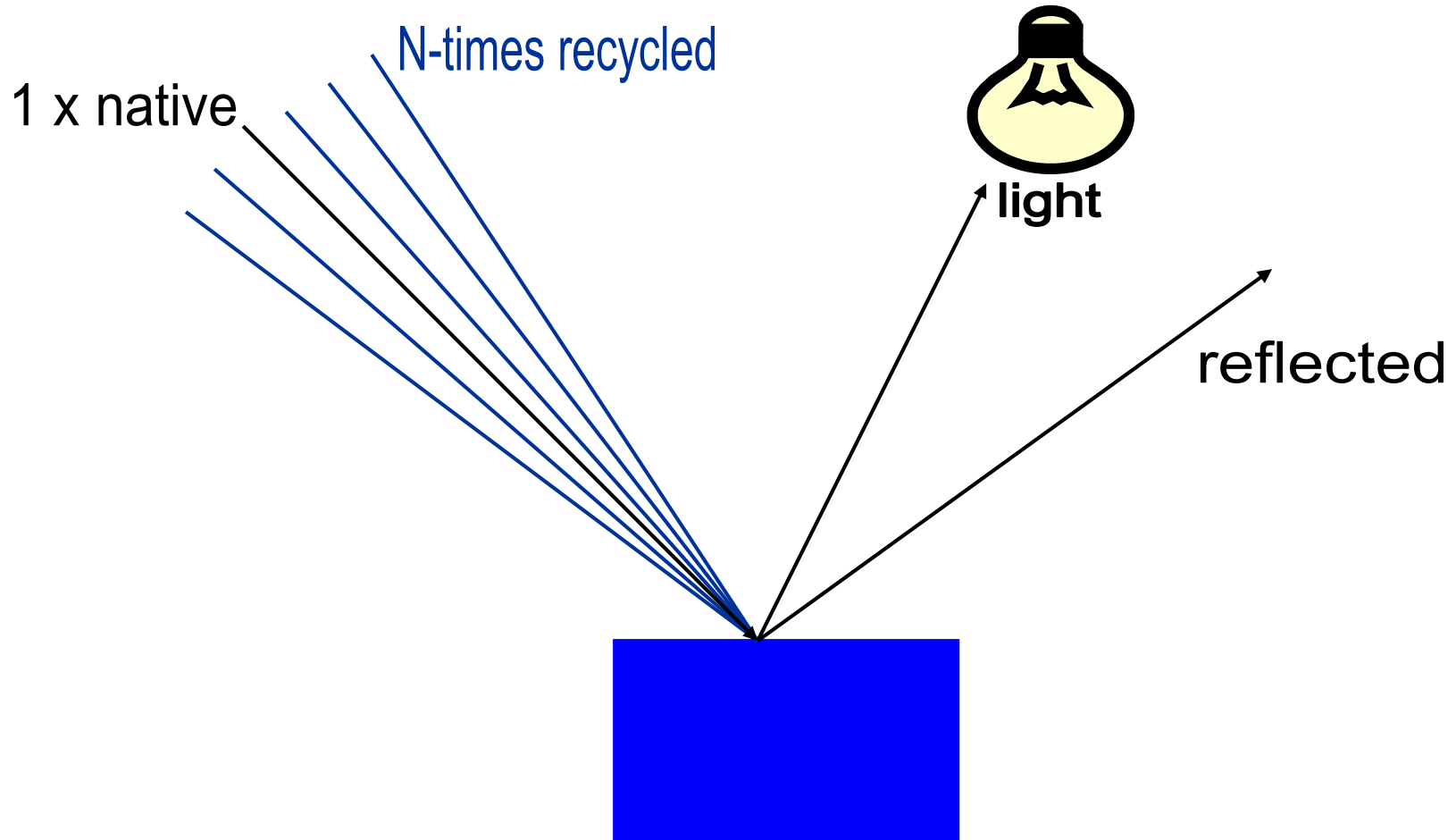
```
for each pixel                                // spatial domain
  while sample variance criterion is met
    compute shaded sample;                   // a sample point in the object space
    for each frame                             // temporal domain
      if possible re-use shaded sample // check visibility and change
        // sample weight
```

Motion Compensation

- Camera and object motion compensation
- Memory access coherence

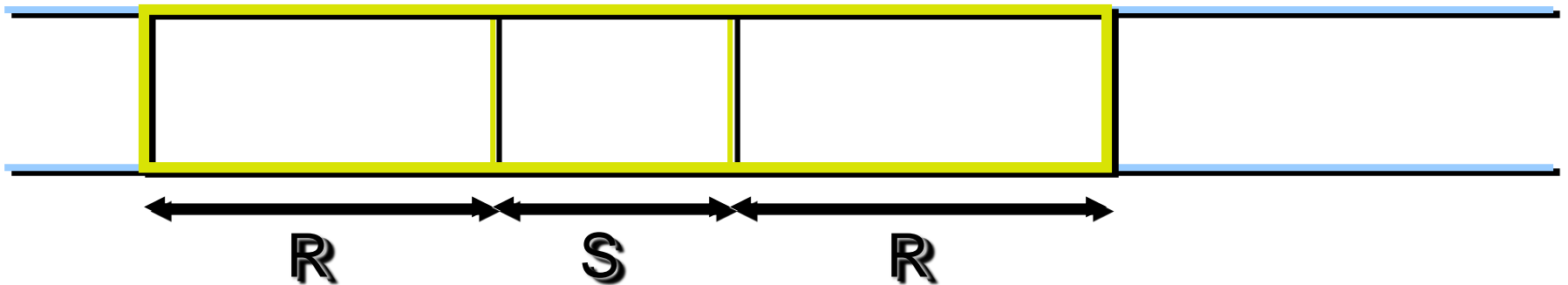


Space-time Architecture: Reprojection



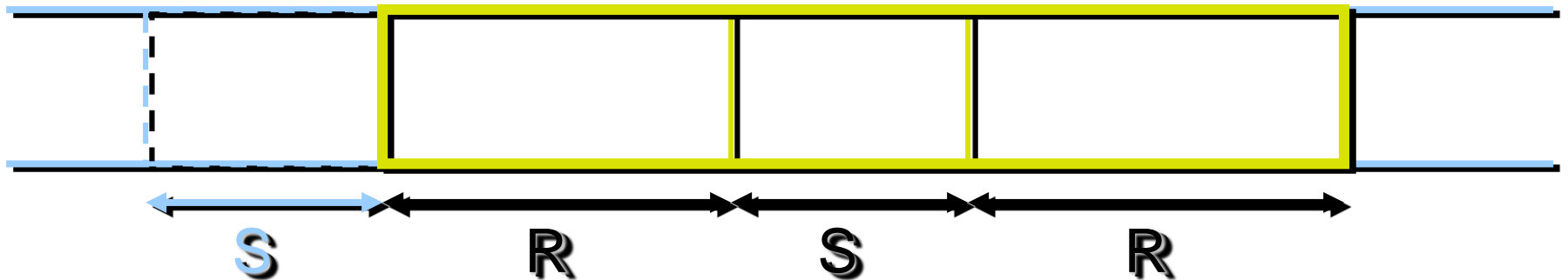
The Animation Buffer

- Iterates over all pixels in S consecutive frames
- If more samples are required
 - Compute a native sample for frame f_i
 - Reproject it and recycle it for all frames in $[f_{(i-R)}, f_{(i+R)}]$
- **$S+2R$ frames are kept in the buffer**

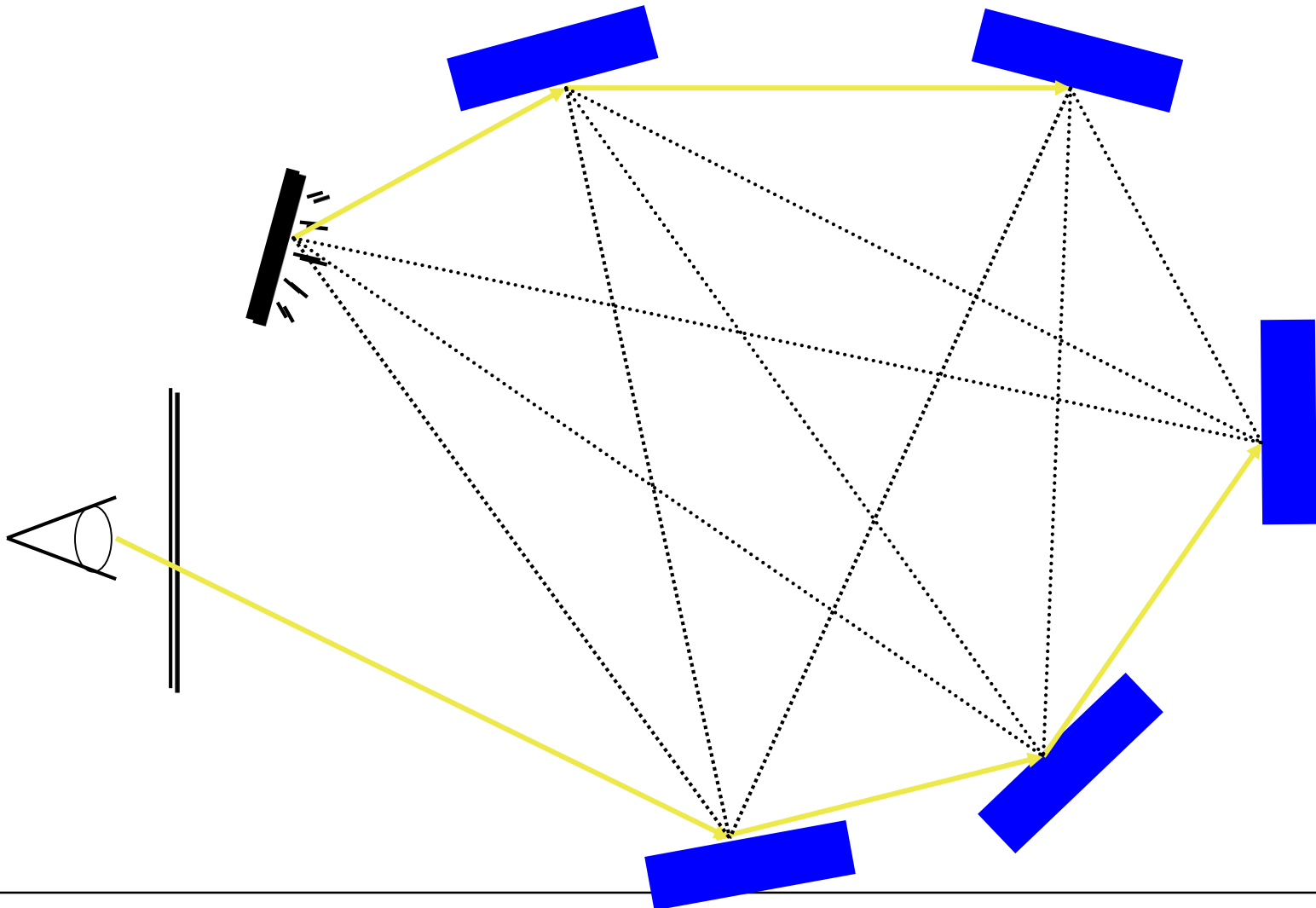


The Animation Buffer

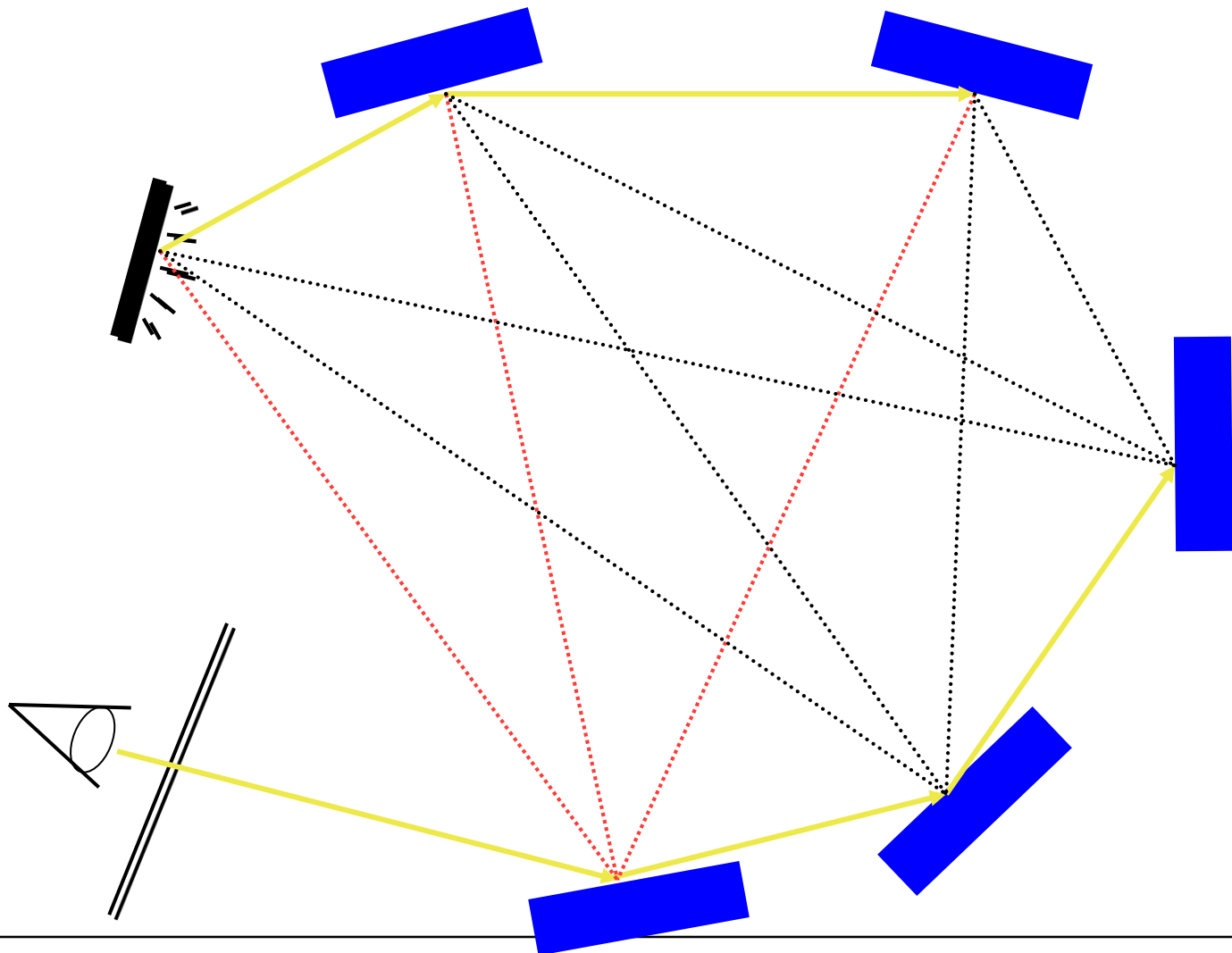
- Iterates over all pixels in S consecutive frames
- If more samples are required
 - Compute a native sample for frame f_i
 - Reproject it and recycle it for all frames in $[f_{(i-R)}, f_{(i+R)}]$
- **$S+2R$ frames are kept in the buffer**



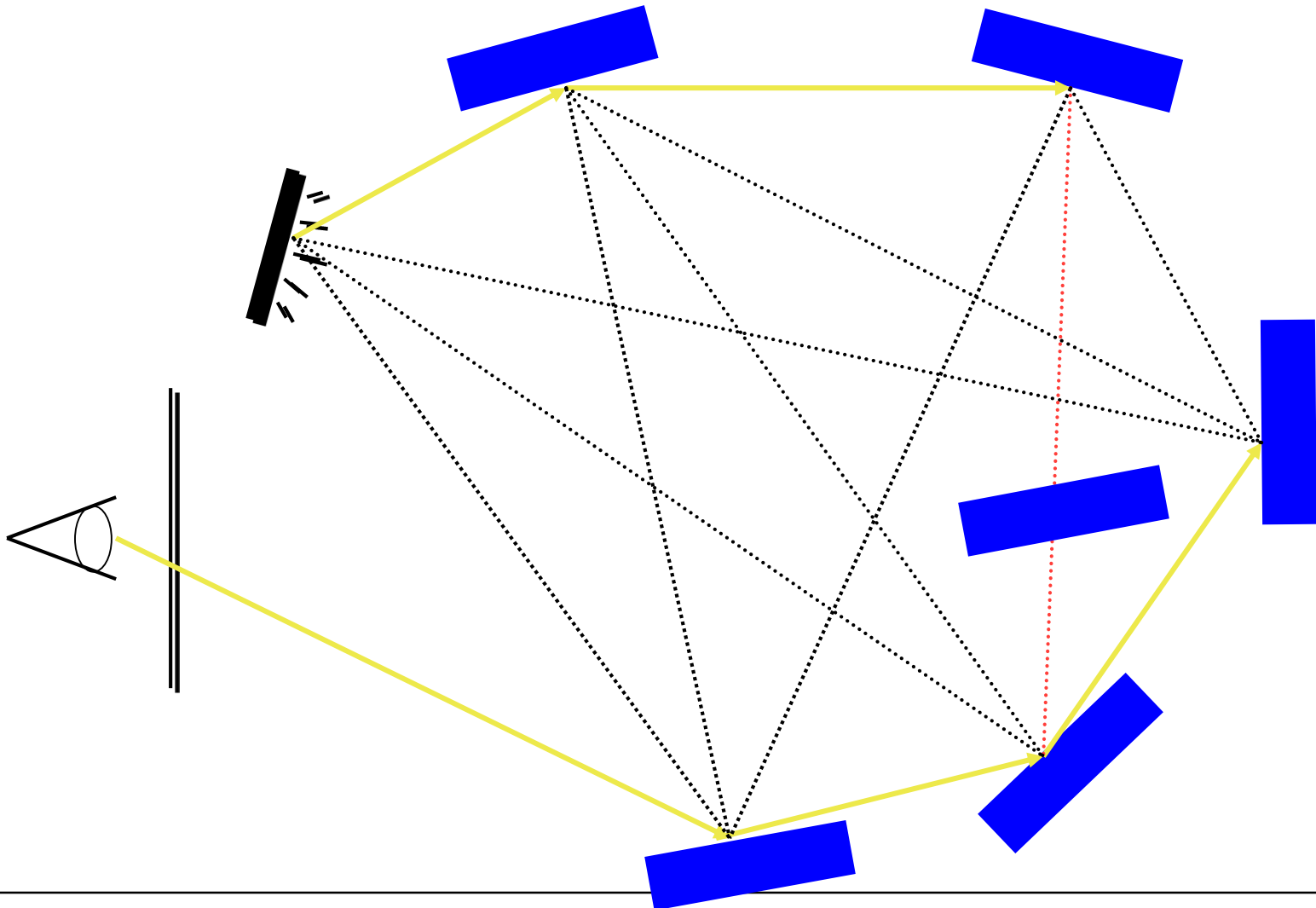
Bi-directional Path Tracing



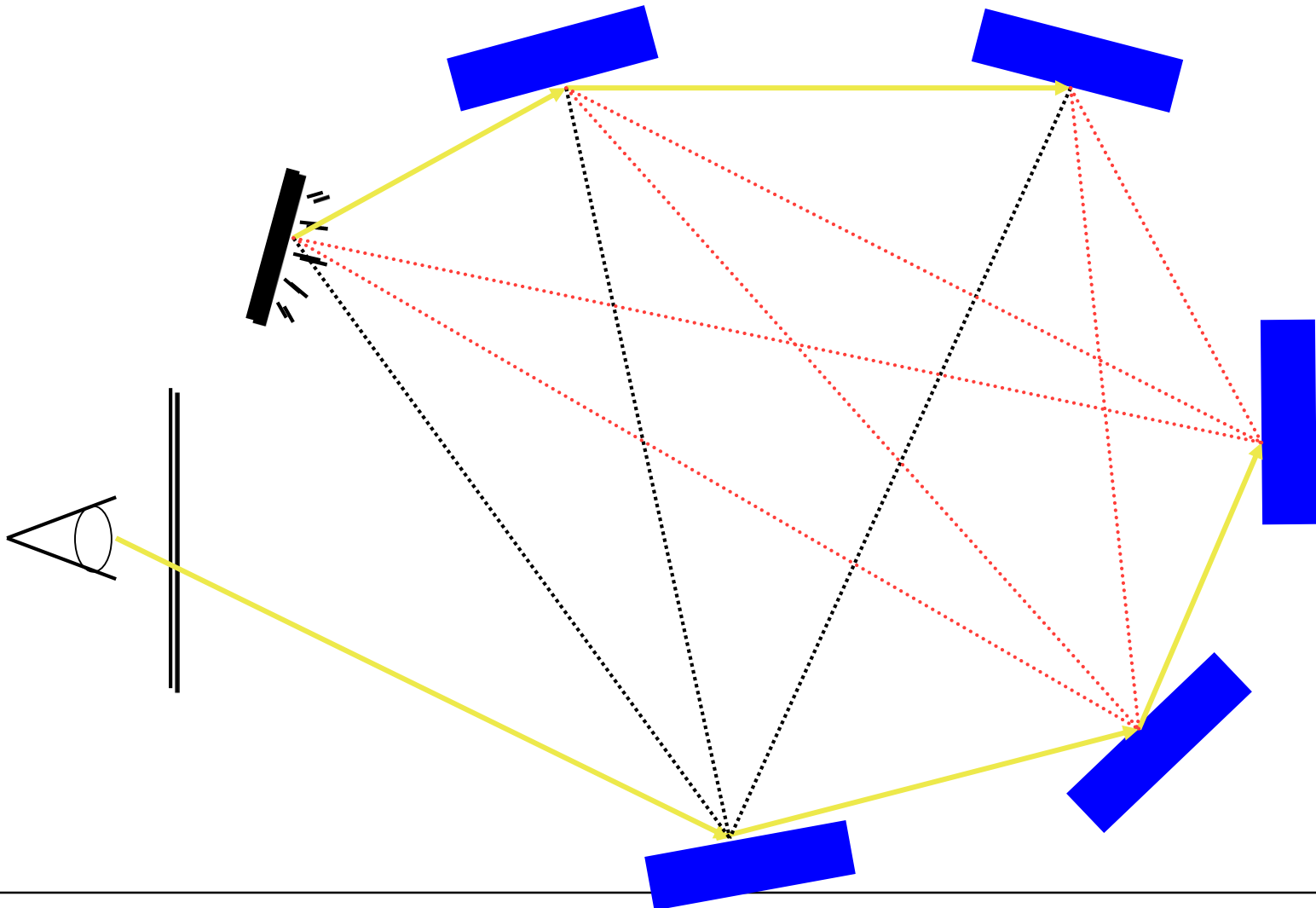
Camera Motion



Occluded Connection



Path Change



Shading Computation

- **A simplified version of RenderMan Shading Language**
- **Each shader decomposed into**
 - View-independent component
 - re-usable, shared between frames
 - View-dependent component
 - recomputed for each frame



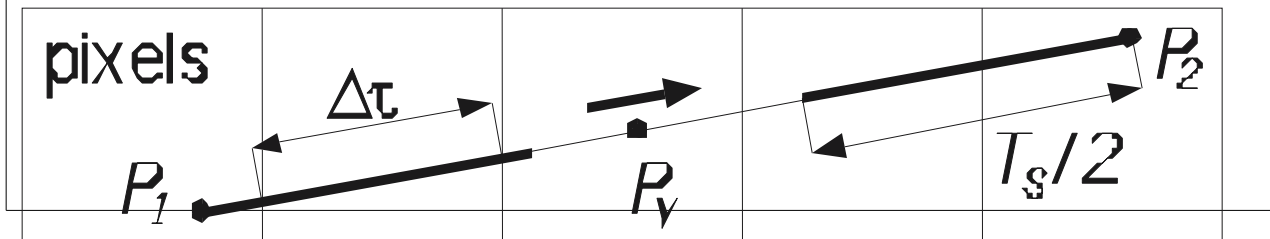
Motion Blur

- **Accuracy & quality**

- The same sample point is considered for multiple frames
 - In other frame-by-frame architectures the motion of objects must be computed explicitly by additional samples.
- Temporal changes in shading are properly accounted for
 - Difficult in other architectures

- **Efficiency**

- 2D computation



Motion Blur Examples



Video: Motion Blur



Video: Traditional Approach



Video: Spatio-temporal Approach



Results

- **Speedup**

- Moving camera, moving objects: 7.7
- Moving objects only: 8.8
- Moving camera only: 13.3



- **Proportion of native samples** 2.4 - 4.7 %
- **Cost of native samples (profiler)** 44 - 64 % of the whole computation time.

Discussion

- + An efficient architecture for rendering of high-quality animations tailored for path tracing algorithms
 - Makes those costly and unbiased algorithms usable for animation at all
- + Temporal flickering substantially suppressed
 - Even noise inherent for path tracing algorithms appears as a static texture assigned to object surfaces
- + Texturing and shading, motion blur can be efficiently handled
- + The memory overhead involved in storing multiple images is negligible due to efficient buffering
- Data structures handling dynamic objects require additional memory
 - Still acceptable on modern computers
- Efficiency may drop significantly for scenes involving too many dynamic objects

Summary

- **Off-line global illumination for animations**
 - Reduction of the rendering cost per frame very important
 - Can be achieved by better exploiting temporal coherence
 - Better performance
 - Better quality - reduced temporal aliasing
- **Successful solutions exist,
but ... still many things to do**
 - Affordable techniques supporting glossy effects
 - Design of an efficient renderer architecture performing the computation directly in the spatio-temporal domain

Motion Blur

- **Important to combat spatio-temporal aliasing**
 - Visibility (geometric) aliasing
 - Shading aliasing
 - Relatively little attention focused on the motion blur in the context of global illumination
- **Desirable to simulate optical systems with controllable shutter speed**
 - Viewer expects to see this effect
 - Required to seamlessly composite virtual and real world shots
- **Main problem**
 - Handling scenes that contain high temporal frequencies
 - Fast moving objects
 - Fast shading changes in time

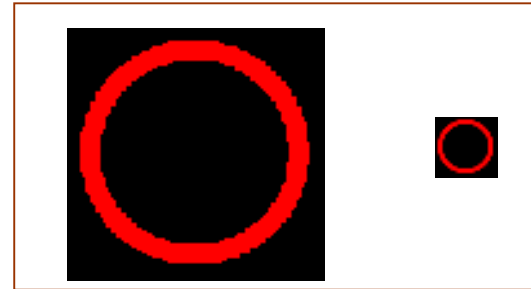


Motion Blur

Brute force solution: supersampling

Spatial domain:

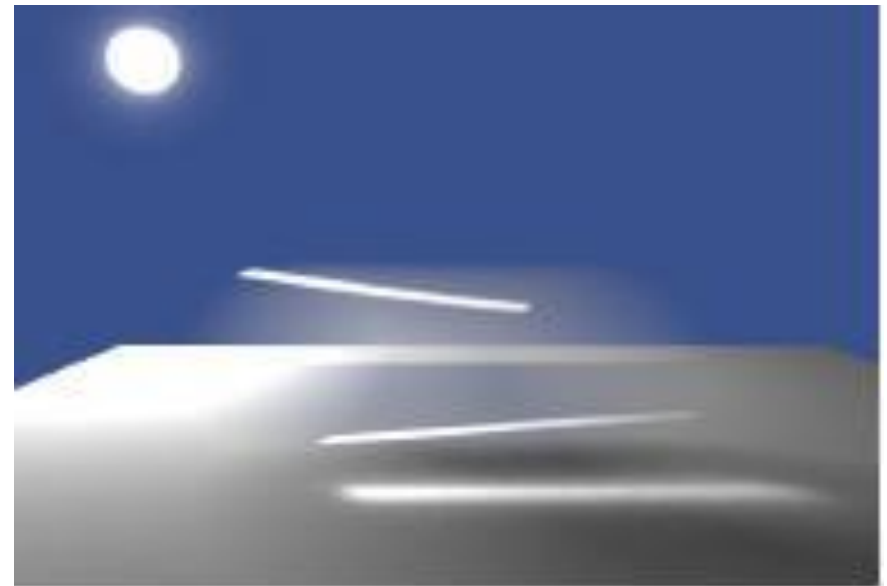
Render a higher resolution image and resample it through averaging neighboring pixels



Supersampling in temporal domain: strobing artifacts



Integrating over the exposure interval: proper motion blur

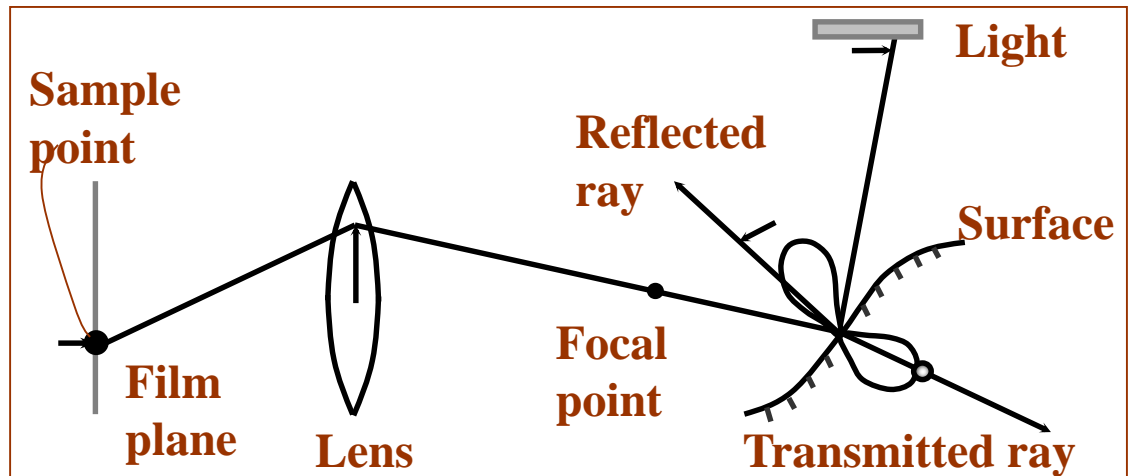


Monte Carlo Integration

- Represent the pixel intensity as a multidimensional Monte Carlo integral and use the ***distribution ray tracing*** to approximate it:

$$i(\omega, t) \approx \frac{1}{N_j} \frac{1}{N_k} \sum_j \sum_k \sum_l r(\omega_j, t_k) g_l(\omega_j, t_k) L_l(\omega_j, t_k)$$

- Use ***bidirectional path tracing*** for motion blur and global illumination computation [Lafortune'96]



Advantage: can handle general $L_l(\omega, t)$ and complex $g(\omega, t)$

Disadvantage: time-consuming to eliminate visible noise

Monte Carlo Integration

Monte Carlo ray tracing

- ⊕ It is the only method that is able to simultaneously simulate both motion blur and global illumination
- ⊖ Gives noisy results and requires extensive computations to reduce the noise below its visibility level
- ⊕ Acceleration techniques: *irradiance caching*, *photon mapping*
- ⊖ These techniques require extensions to make them working for dynamic scenes and handling time dependent effects

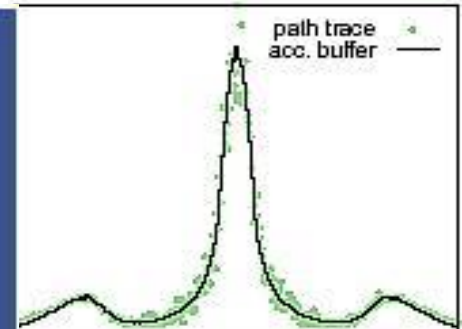
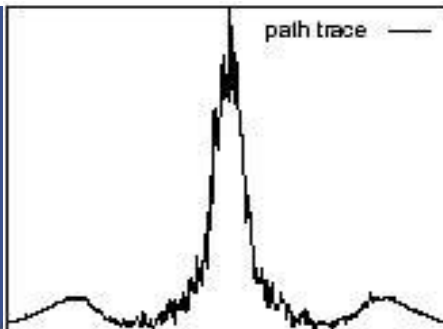
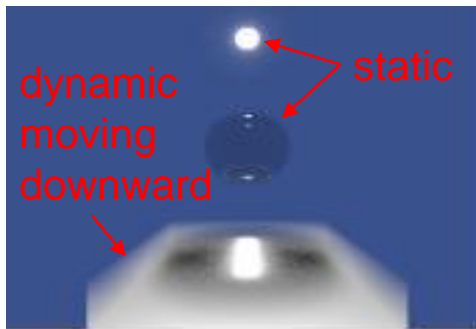
Solution:

Extend photon mapping technique with *time dependent radiance estimate*

Time Dependent Photon Mapping

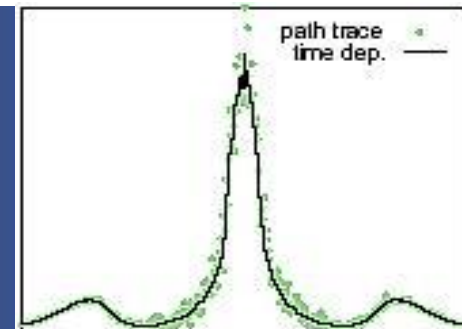
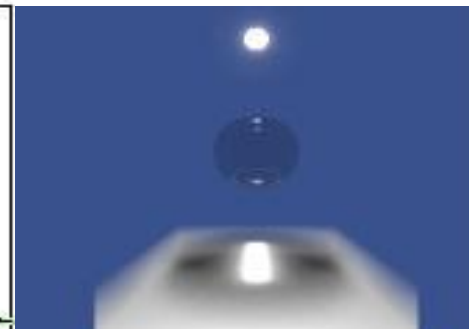
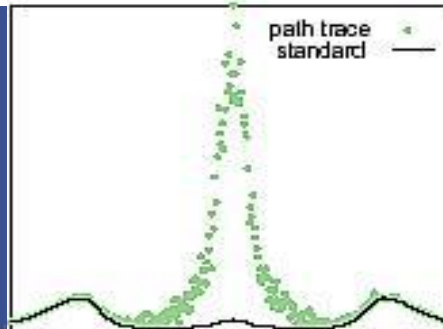
- **Trace photons distributed in time and space**
 - Include time information to the photon data structure
- **Lighting reconstruction**
 - For localizing spatially adjacent photons use 3D kd-tree search
 - Select the nearest of those photons in time using a randomized quicksort
 - This requires locating 50% more photons than in the standard technique
- **Localizing photons using a 4D kd-tree**
 - More complicated and less elegant
- **Only motion blur of $L(\omega, t)$ is directly handled**
 - Distribution ray tracing is required to handle temporal changes in visibility $g(\omega, t)$

Time Dependent Photon Mapping



path tracing (10,000 paths per pixel random in time)

accumulation buffer (20 frames)

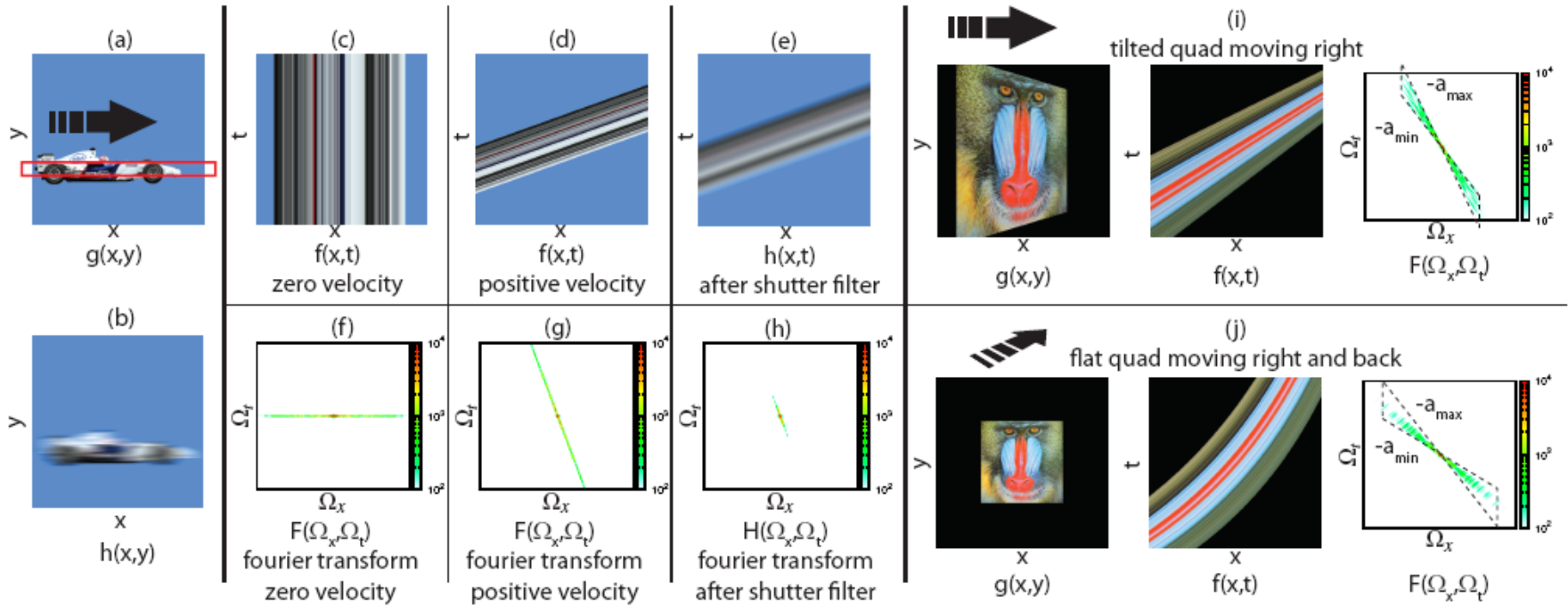


standard radiance estimate

time dependent radiance estimate

Method	Consistent ?	Render times*	
		This scene	Next slide
<i>Path tracing</i>	Yes	9+hrs.	n/a
<i>Accumulation buffer</i>	Yes	47 sec	316 sec
<i>Standard radiance estimate</i>	No	37 sec	74 sec
<i>Time dependent radiance estimate</i>	Yes	43 sec	72 sec

Sheared Reconstruction for Motion Blur

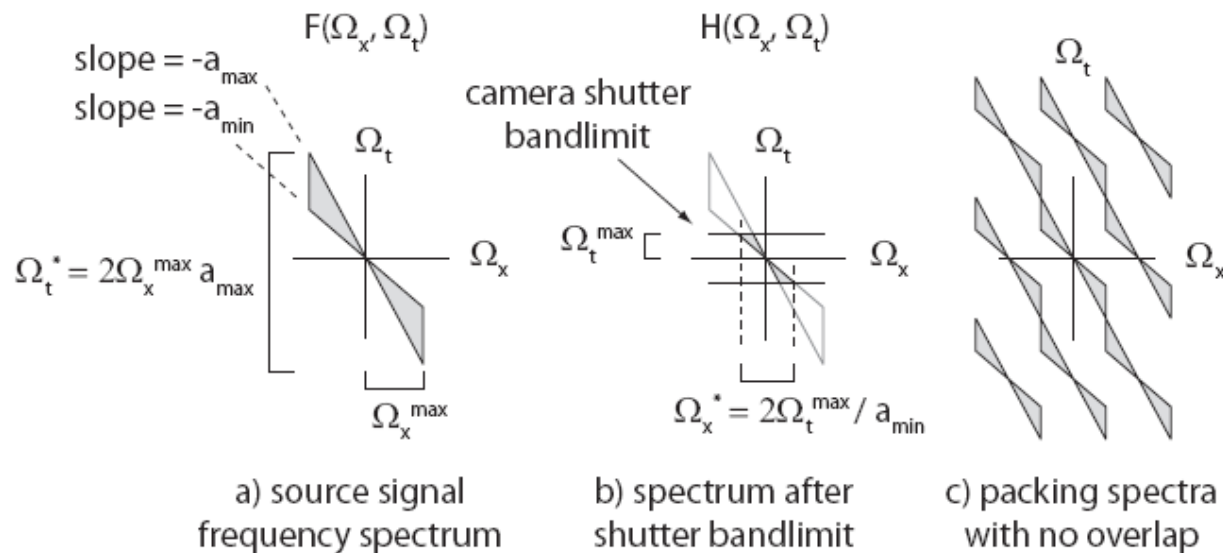


Space-Time and Fourier domain plots for a moving object. (a) Original signal $g(x,y)$; the scanline used for graphs (c), (d), and (e) is outlined in red. (b) (below (a)) $h(x,y,t)$ for a single instant in time; this is our final motion-blurred image. (c) A graph of $f(x,t)$ with zero velocity (a static image). In this case, there is no variation along the time or vertical axis. (d) $f(x,t)$ with positive uniform velocity, leading to a shearing along the spatial dimension. (e) $h(x,y,t)$ is obtained by applying a vertical blur along the time axis corresponding to the shutter filter. (f), (g) and (h) are the respective Fourier transforms of (c), (d) and (e). Note that (h) has frequencies in time restricted to $\Omega_t \in [-\Omega_t^{\max}, \Omega_t^{\max}]$ based on the shutter filter. (i) Because of perspective, the velocities change across space. (j) Because of perspective, velocities change across time. The frequency spectra span a wedge based on the minimum and maximum velocities.

$$F(\Omega_x, \Omega_t) = G(\Omega_x)\delta(\Omega_x a + \Omega_t), \text{ Fourier transform for an image } g(x,y) \text{ moving with a constant velocity } a$$

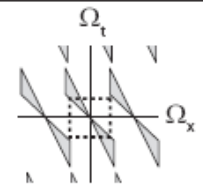
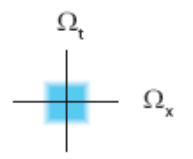
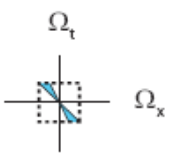
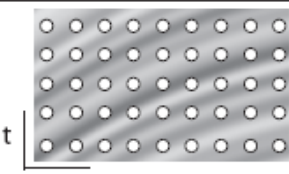
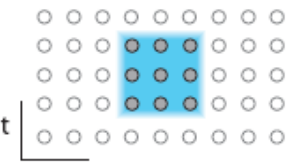
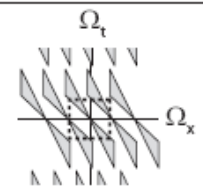
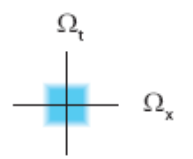
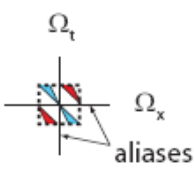
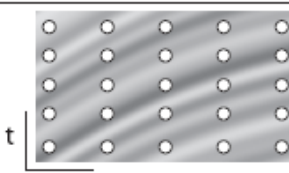
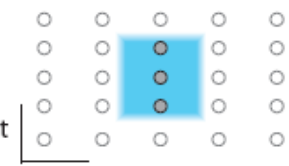
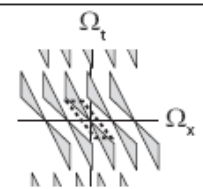
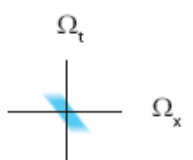
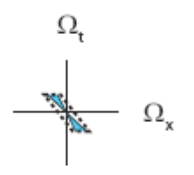
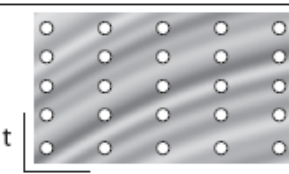
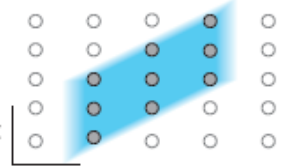
$$H(\Omega_x, \Omega_t) = G(\Omega_x)\delta(\Omega_x a + \Omega_t)W(\Omega_t). \leftarrow W() \text{ is the frequency spectrum of the low-pass shutter filtering}$$

Sheared Reconstruction for Motion Blur



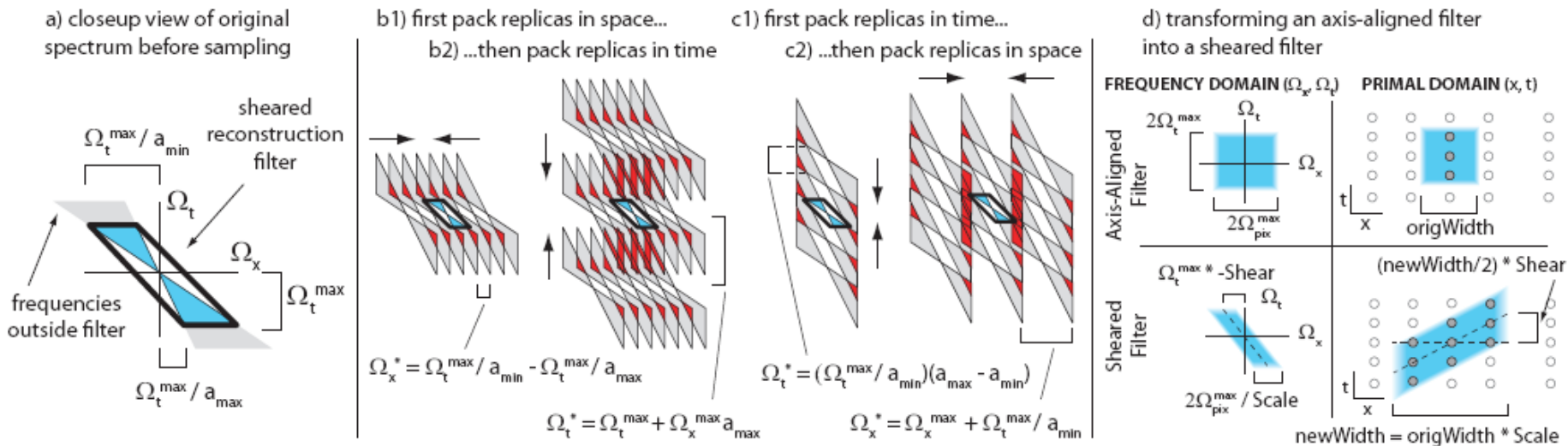
(a) Frequency spectrum of source signal $F(\Omega_x, \Omega_t)$ in space and time (Ω_x and Ω_t). We also mark the highest spatial frequency Ω_x^{\max} , and the highest temporal frequency Ω_t^* , determined by the maximum velocity/shear a_{\max} . (b) The signal is bandlimited in time based on the camera shutter to temporal frequencies less than Ω_t^{\max} . For images with medium to large amounts of motion blur, the spatial frequencies are also correspondingly filtered to Ω_x^* , depending on the minimum velocity a_{\min} . (c) Sampling introduces replicas of the base spectrum F . To achieve a low sampling rate we must bring the spectra as close as possible without aliasing.

Sheared Reconstruction for Motion Blur

	FREQUENCY DOMAIN (Ω_x, Ω_t)			PRIMAL DOMAIN (x, t)	
	1. sampling	2. filtering and reconstruction	3. final result	1. sampling	2. filtering and reconstruction
Method A: dense sampling axis aligned filter NO ALIASING	 <p>dense sampling produces sparse replicas</p>	 <p>axis-aligned reconstruction filter</p>	 <p>signal is bandlimited with no aliasing</p>	 <p>dense sampling moving signal</p>	 <p>axis-aligned filter no aliasing</p>
Method B: sparse sampling axis aligned filter ALIASING	 <p>sparse sampling produces dense replicas</p>	 <p>axis-aligned reconstruction filter</p>	 <p>reconstructed signal has aliasing</p>	 <p>sparse sampling moving signal</p>	 <p>axis-aligned filter with aliasing</p>
Method C: sparse sampling sheared filter NO ALIASING	 <p>sparse sampling produces dense replicas</p>	 <p>sheared reconstruction filter</p>	 <p>signal is bandlimited with no aliasing</p>	 <p>sparse sampling moving signal</p>	 <p>sheared filter, no aliasing</p>

1. Sampling in the primal domain creates replicas in the frequency domain. The denser the sampling the further apart the Fourier-domain replicas are spaced. 2. The Fourier transform of the spatial reconstruction filter bandlimits and reconstructs the signal for display. 3. Filtering the samples in the primal domain is equivalent to multiplying the Fourier transforms of steps 1 and 2. If replicas overlap with the Fourier domain reconstruction filter, the final result would contain spurious frequencies (aliasing). In Method/Row A, a relatively dense sampling is used in space and time to separate the Fourier domain replicas. Method/Row B shows that a sparser sampling rate with an axis-aligned filter leads to aliasing. Method/Row C shows that using a sheared reconstruction filter, we can reconstruct a correct image using a sparse sampling rate.

Sheared Reconstruction for Motion Blur



(a) Zoomed-in view of the frequency wedge and the sheared reconstruction filter. The distances between the Ω_t axis and the near and far points of the sheared filter are shown. Only the blue frequency content inside of the sheared reconstruction filter will be output for display. (b1) and (b2) show packing of replicas as tightly as possible first in space, then in time. (c1) and (c2) show packing of replicas as tightly as possible first in time, then in space. (d) Transforming an axis-aligned filter into a sheared filter. (d Top) We start with any standard axis-aligned filter in the frequency and primal domains. (d Bottom) We then consider the scale and shear in the frequency domain, applying the opposite scale and shear in the space-time domain (Equations 33 and 34).

$$\text{Shear} = \frac{1}{2} \left(\frac{1}{a_{\max}} + \frac{1}{a_{\min}} \right)$$

The **shear** corresponds to the direction of average motion in the space-time domain, with motion compensating filtering (the filter “following the motion”).

$$\text{Scale} = \left(\frac{\Omega_t^{\max}}{2\Omega_x^{\max}} \left(\frac{1}{a_{\min}} - \frac{1}{a_{\max}} \right) \right)^{-1}$$

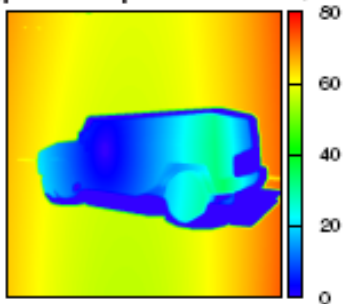
The **scale** depends on the complexity of motion – the filter is larger the closer the maximum and minimum velocities a_{\max} and a_{\min} are.

Sheared Reconstruction: Algorithm

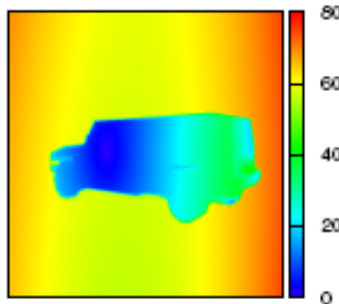
- 1. Compute bounds for signal speeds and spatial frequency**
- 2. Locally decide on the filter shape and sampling density**
- 3. Compute samples and reconstruct image**

#1: Bounds for Signal Speeds and Spatial Frequency

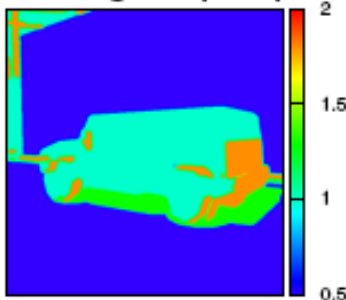
(a) a_{\min} min speed
(pixels per frame)



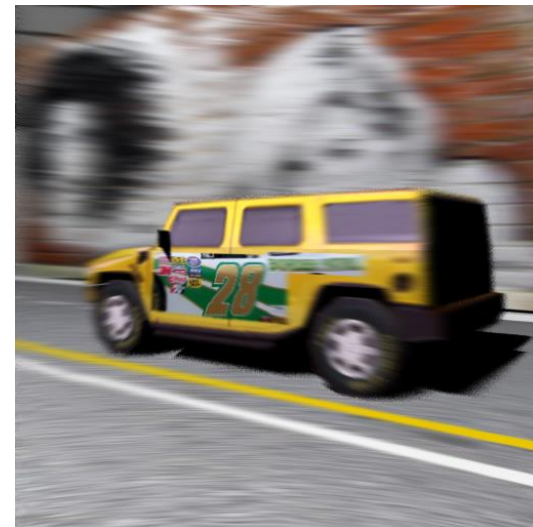
(b) a_{\max} max speed
(pixels per frame)



(c) Ω_x^{\max} spatial frequency
(wavelengths per pixels)



motion blurred

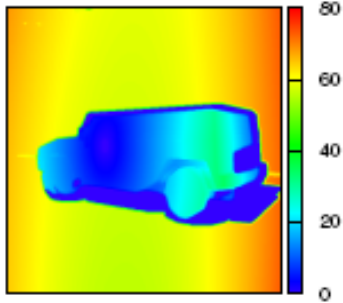


static

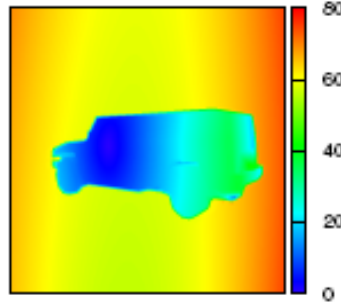


#2: Filter Shape and Sampling Density

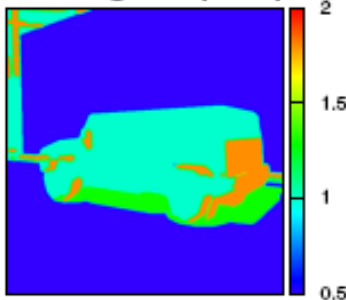
(a) a_{\min} min speed
(pixels per frame)



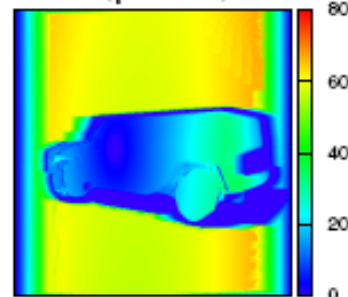
(b) a_{\max} max speed
(pixels per frame)



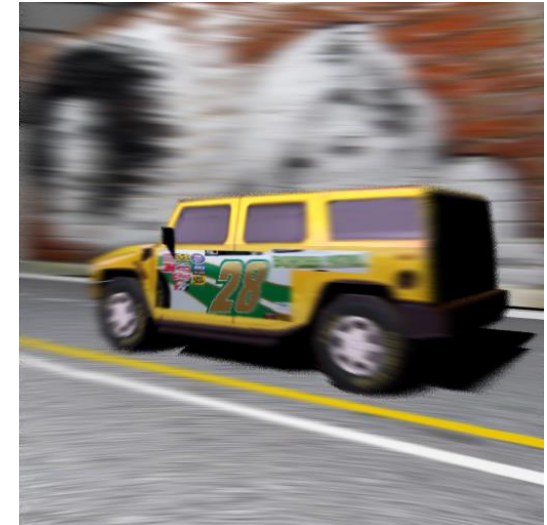
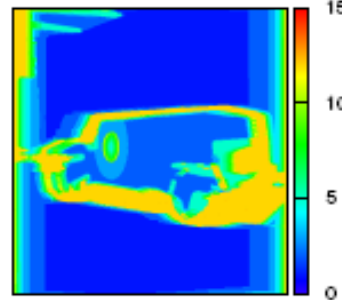
(c) Ω_x^{\max} spatial frequency
(wavelengths per pixels)



(d) filter width in
(pixels)

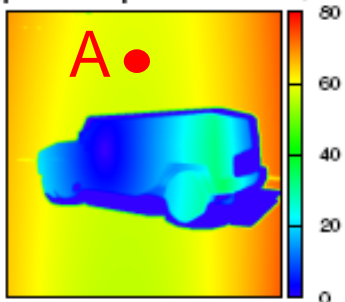


(e) sampling density
(samples per pixel)

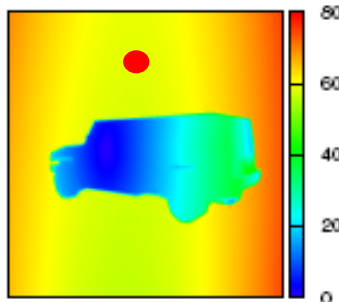


#3: Final Reconstruction

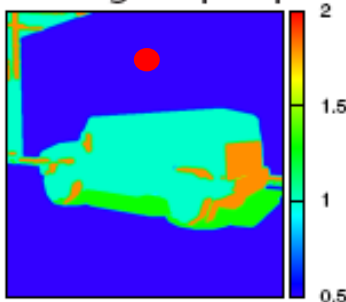
(a) a_{\min} min speed
(pixels per frame)



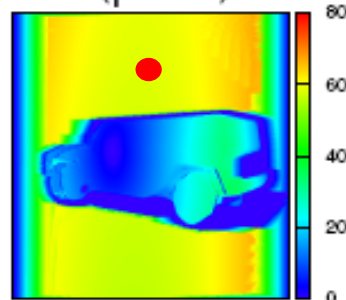
(b) a_{\max} max speed
(pixels per frame)



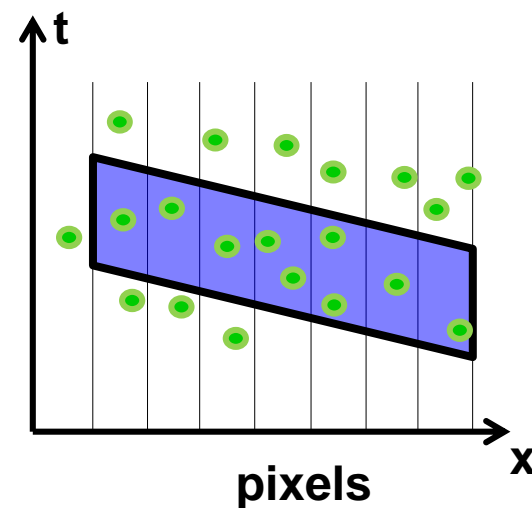
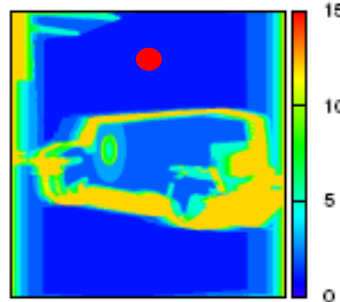
(c) Ω_x^{\max} spatial frequency
(wavelengths per pixels)



(d) filter width in
(pixels)



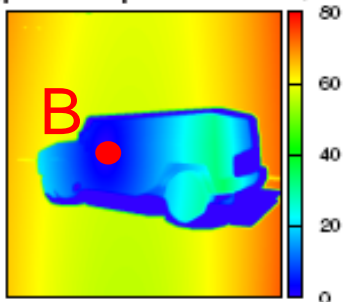
(e) sampling density
(samples per pixel)



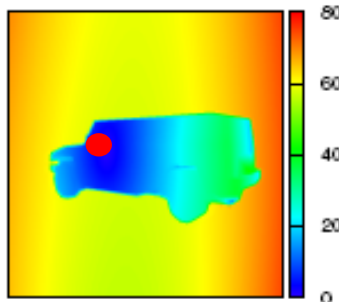
Background A: Uniform velocities, wide filter, low samples

#3: Final Reconstruction

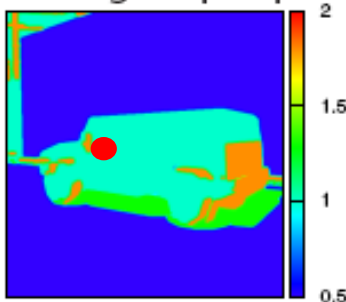
(a) a_{\min} min speed
(pixels per frame)



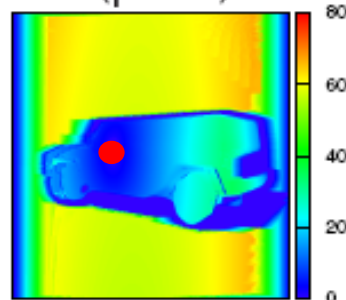
(b) a_{\max} max speed
(pixels per frame)



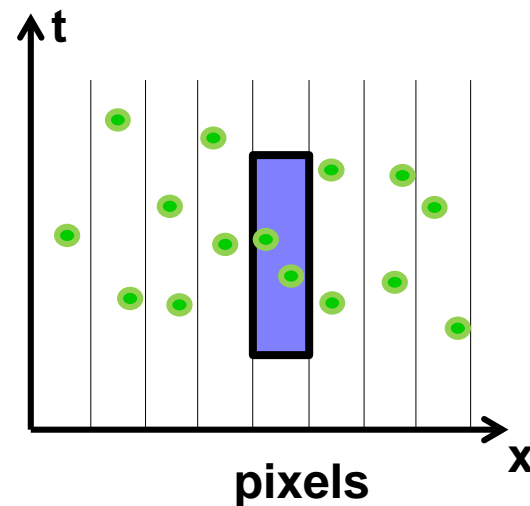
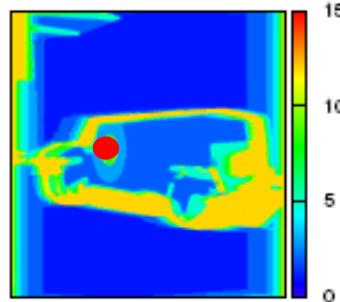
(c) Ω_x^{\max} spatial frequency
(wavelengths per pixels)



(d) filter width in
(pixels)



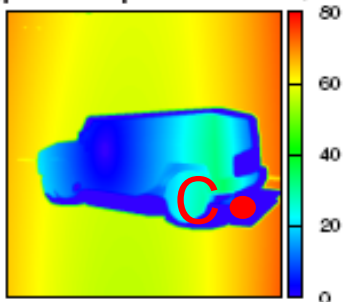
(e) sampling density
(samples per pixel)



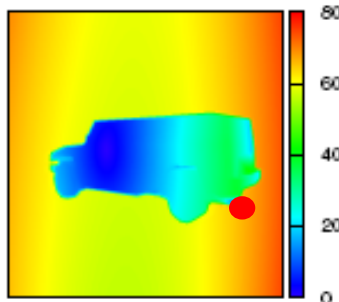
Car B: static region, small filter, low sample density

#3: Final Reconstruction

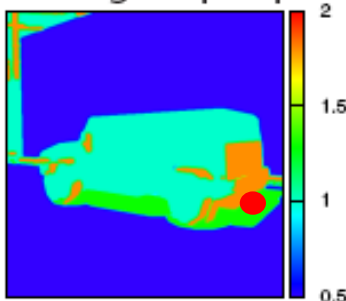
(a) a_{\min} min speed
(pixels per frame)



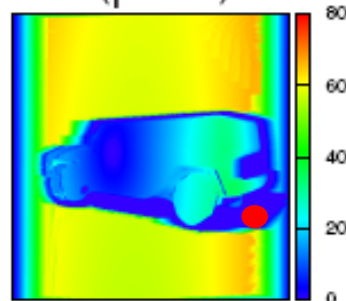
(b) a_{\max} max speed
(pixels per frame)



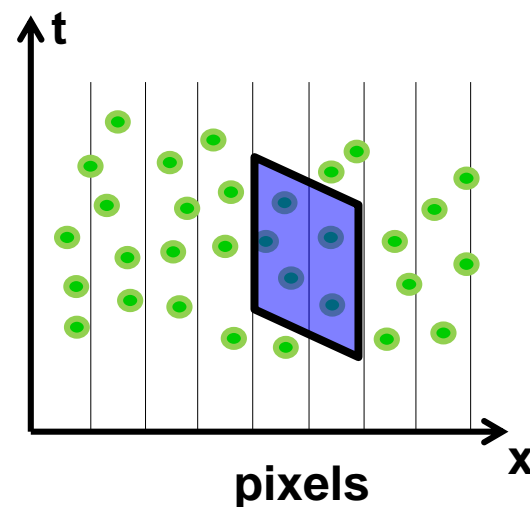
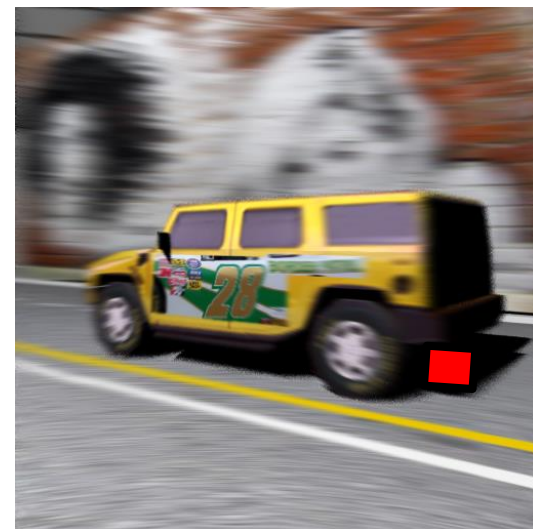
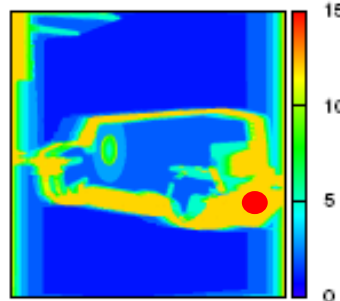
(c) Ω_x^{\max} spatial frequency
(wavelengths per pixels)



(d) filter width in
(pixels)

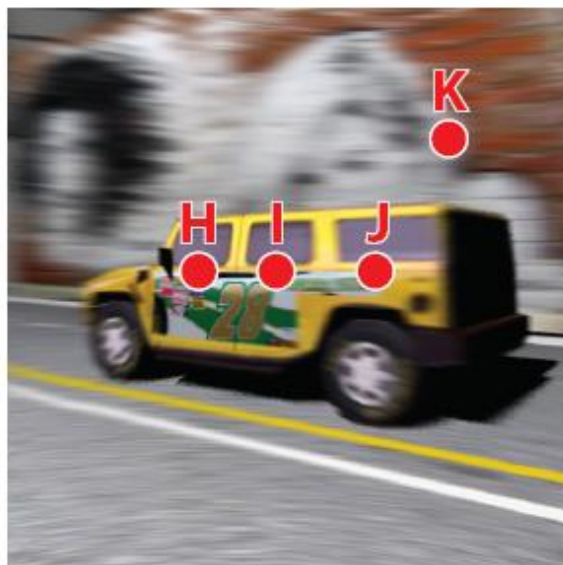


(e) sampling density
(samples per pixel)



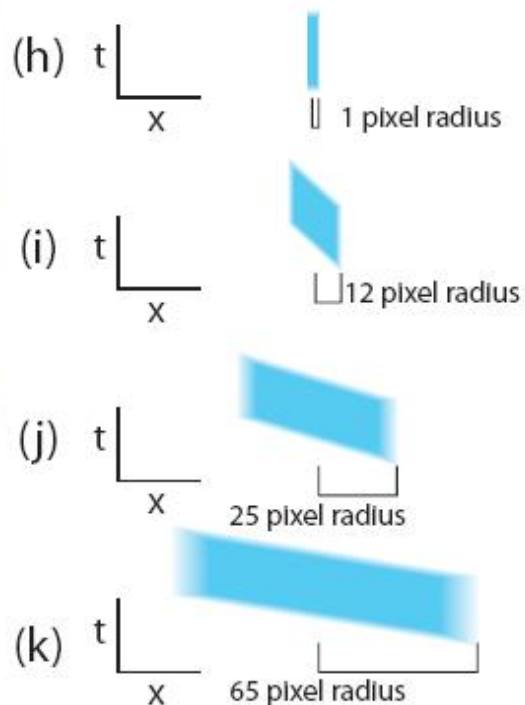
Shadow C: Varying velocities, small filter, high sample density

Final Reconstruction: Summary



- (h) static surface, axis-aligned filter
- (i) moving surface, sheared filter
- (j) moving surface, sheared filter
- (k) moving surface, sheared filter

reconstruction filter shapes



- **Filters stretched along direction of motion**
- **Preserve frequencies orthogonal to motion**