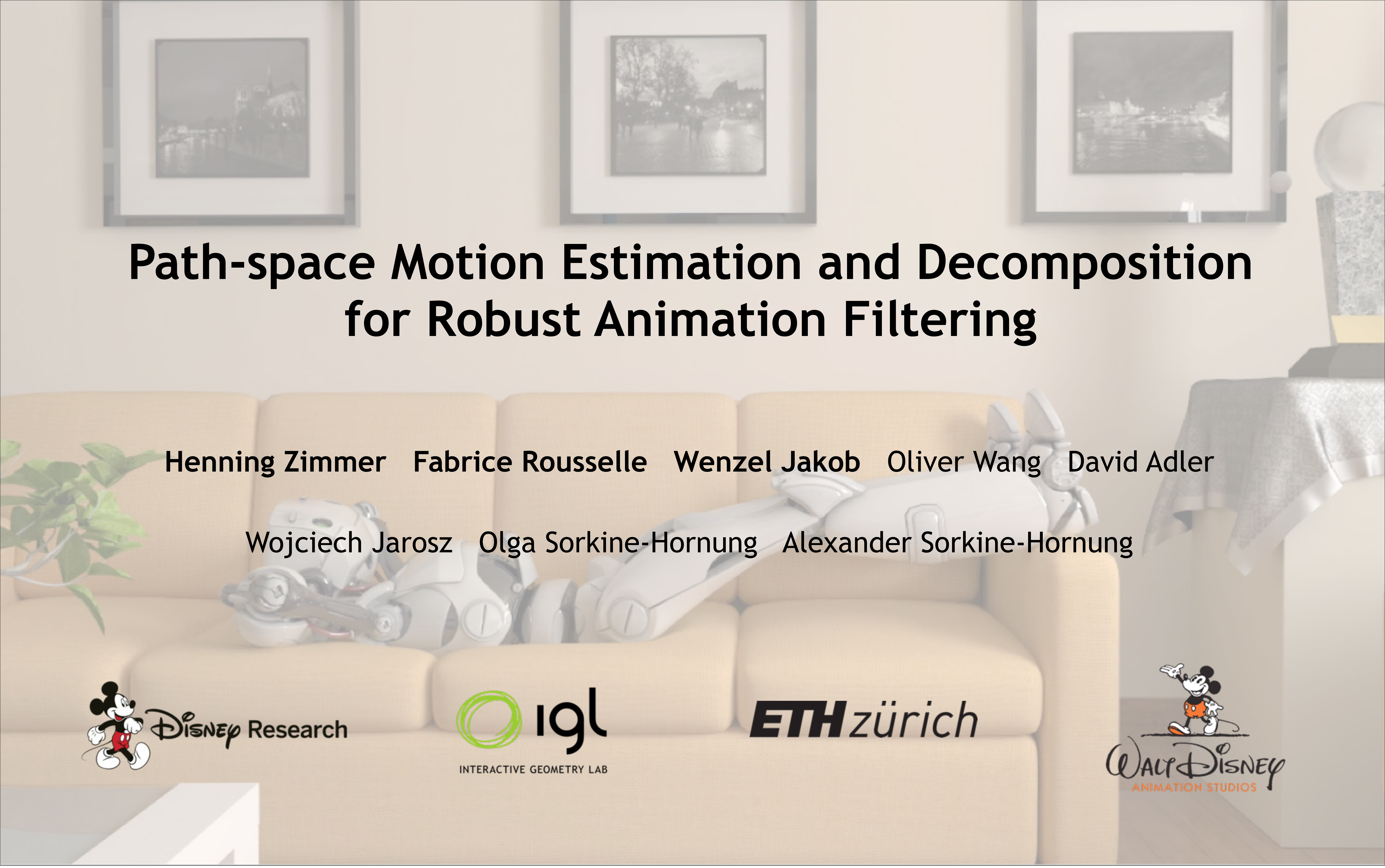


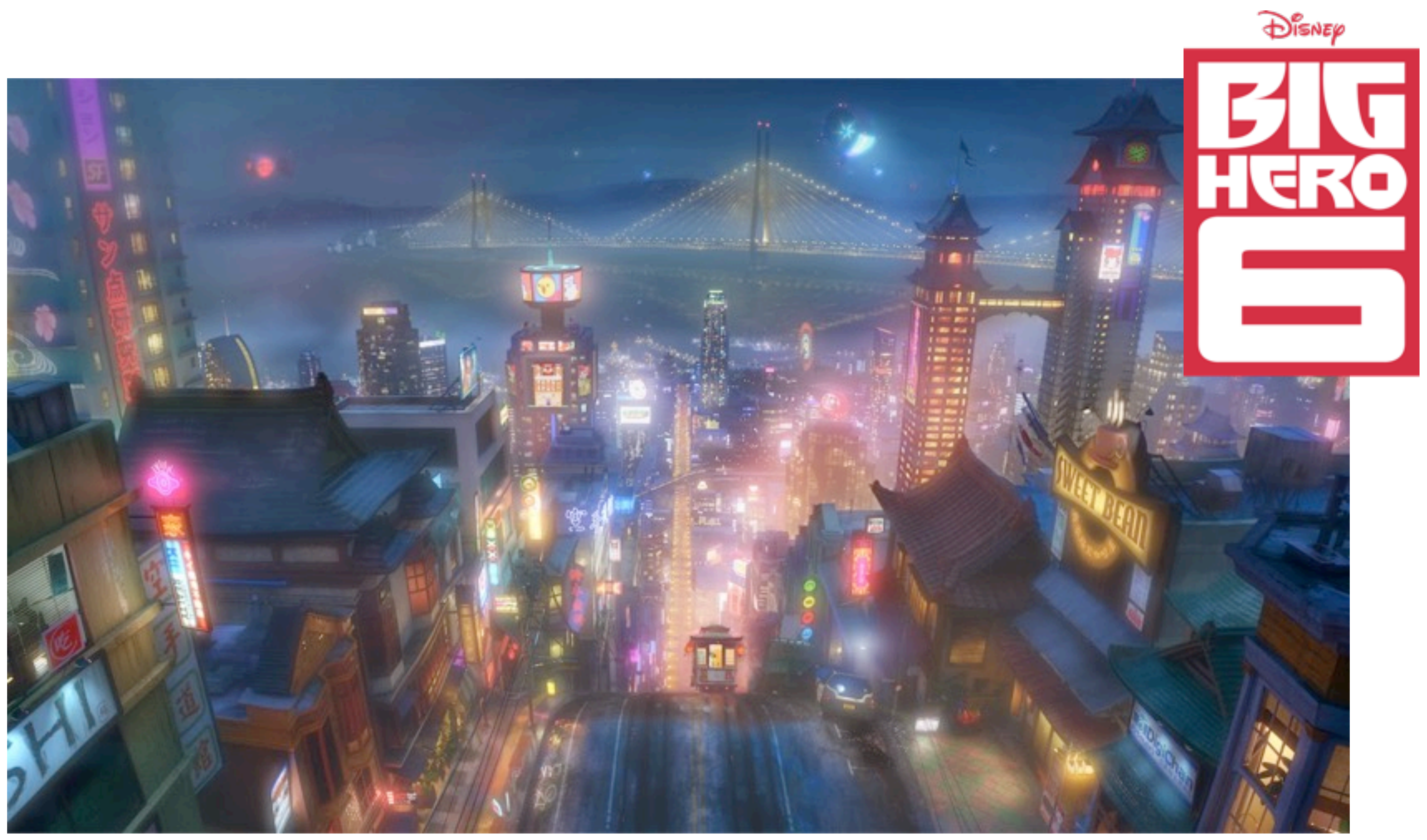
Path-space Motion Estimation and Decomposition for Robust Animation Filtering

Henning Zimmer **Fabrice Rousselle** **Wenzel Jakob** Oliver Wang David Adler

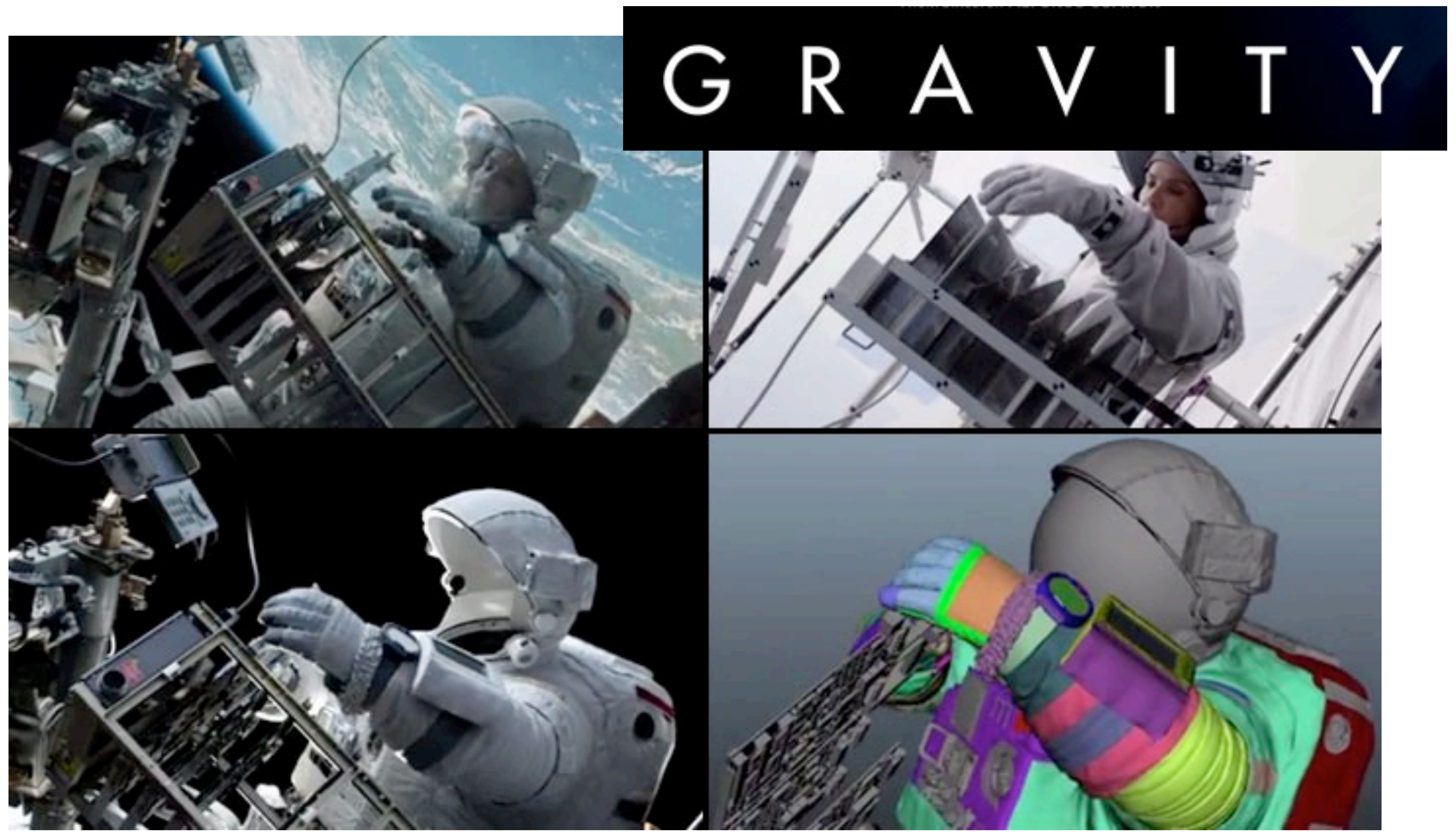
Wojciech Jarosz Olga Sorkine-Hornung Alexander Sorkine-Hornung

Monday, June 29, 15

Motivation



Feature Animation



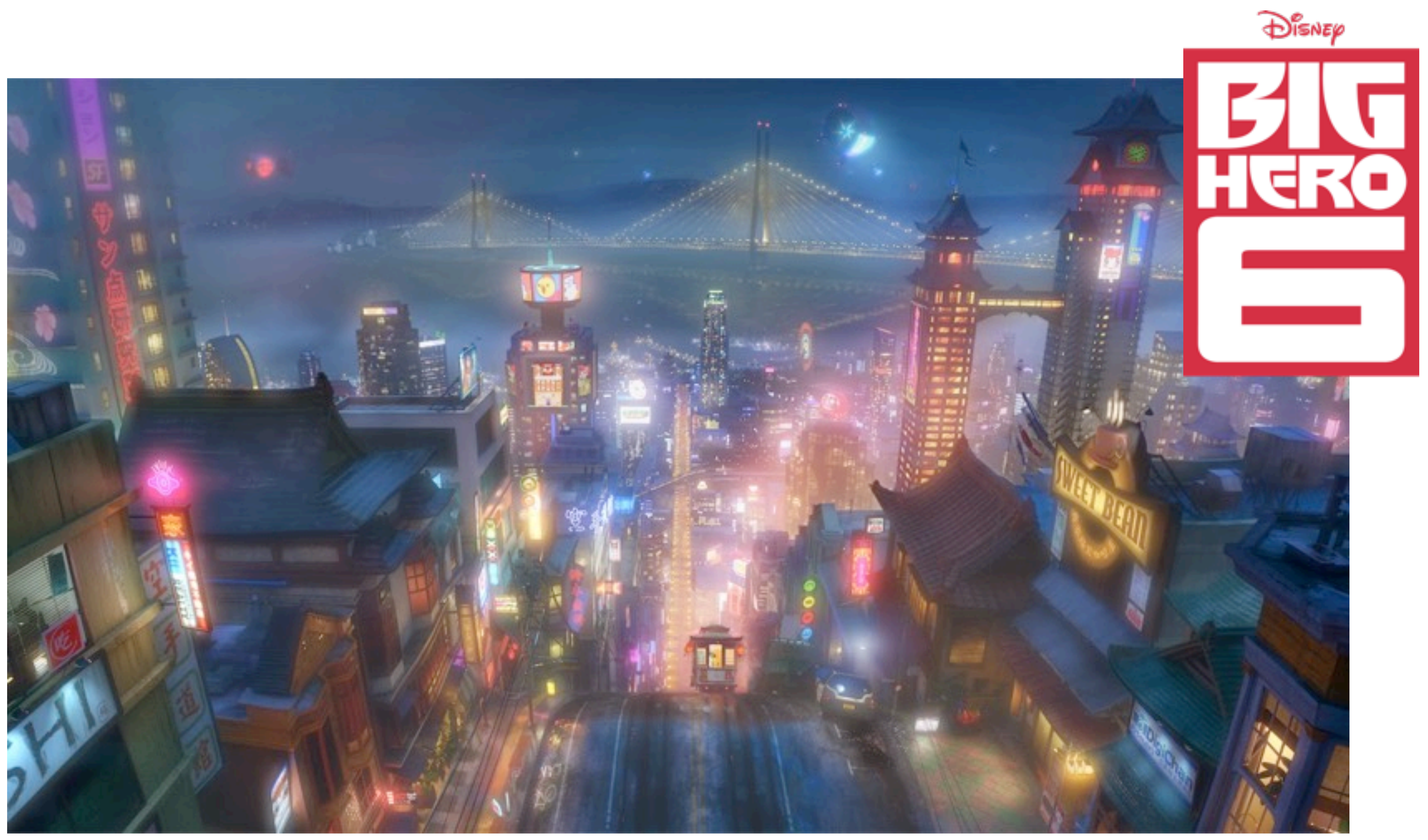
Visual Effects

This is a really exciting time for rendering with Monte Carlo techniques now seeing a wide adoption in the animation and VFX communities. But this trend has created tremendous problems: <click>

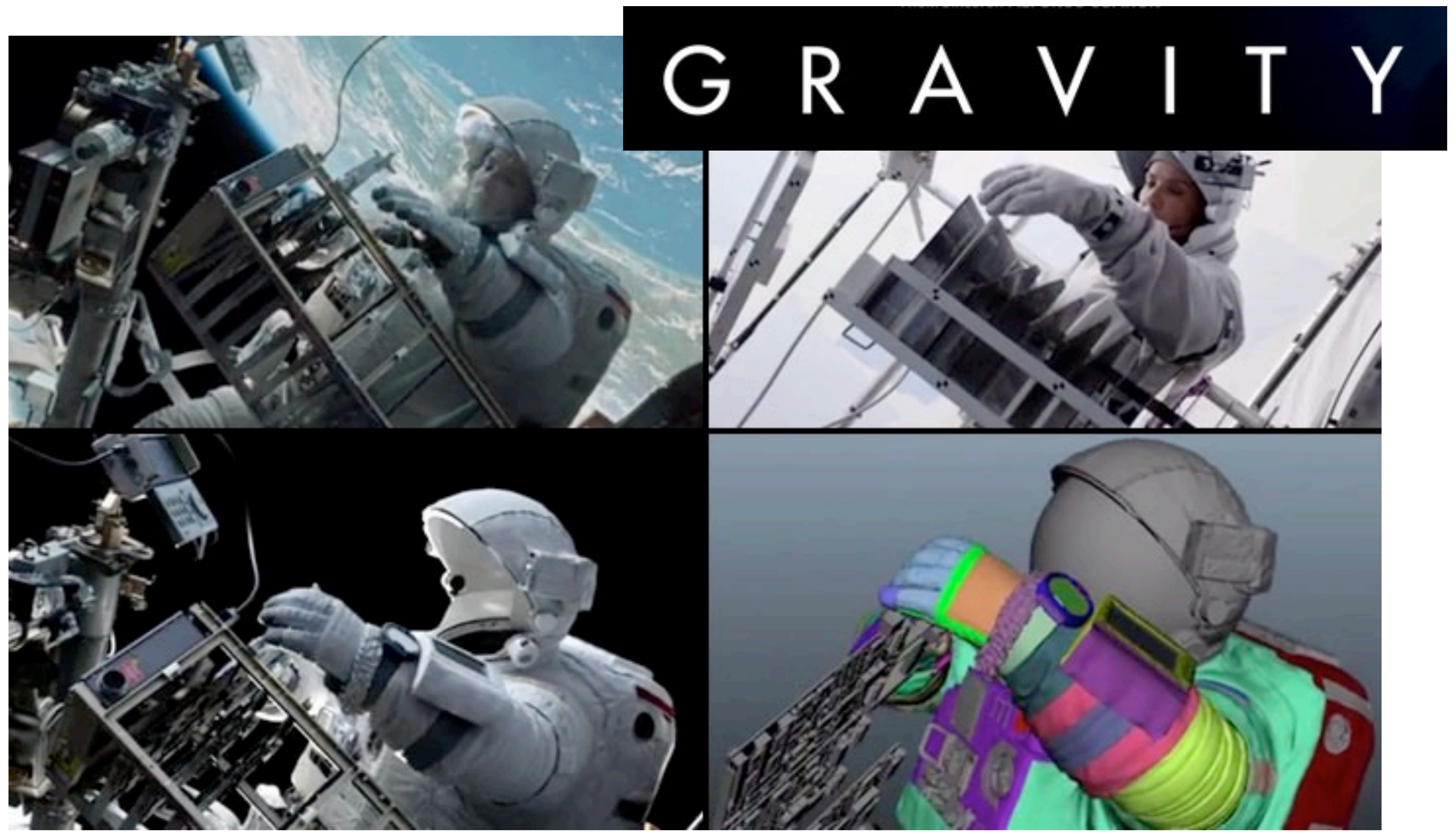
people want 4K renderings at high frame rate, and in stereo. The sheer amount of pixels that need to be rendered has become a significant computational burden, since these Monte Carlo techniques are just not fast enough.

But there is so much coherence in space and time that we can in practice often render the data sparsely and recover the final image using image-based post-processing techniques.

Motivation



Feature Animation



Visual Effects

Stereo

High resolution

High frame rate

Disney Research igl ETH zürich Walt Disney Animation Studios

This is a really exciting time for rendering with Monte Carlo techniques now seeing a wide adoption in the animation and VFX communities. But this trend has created tremendous problems: <click>

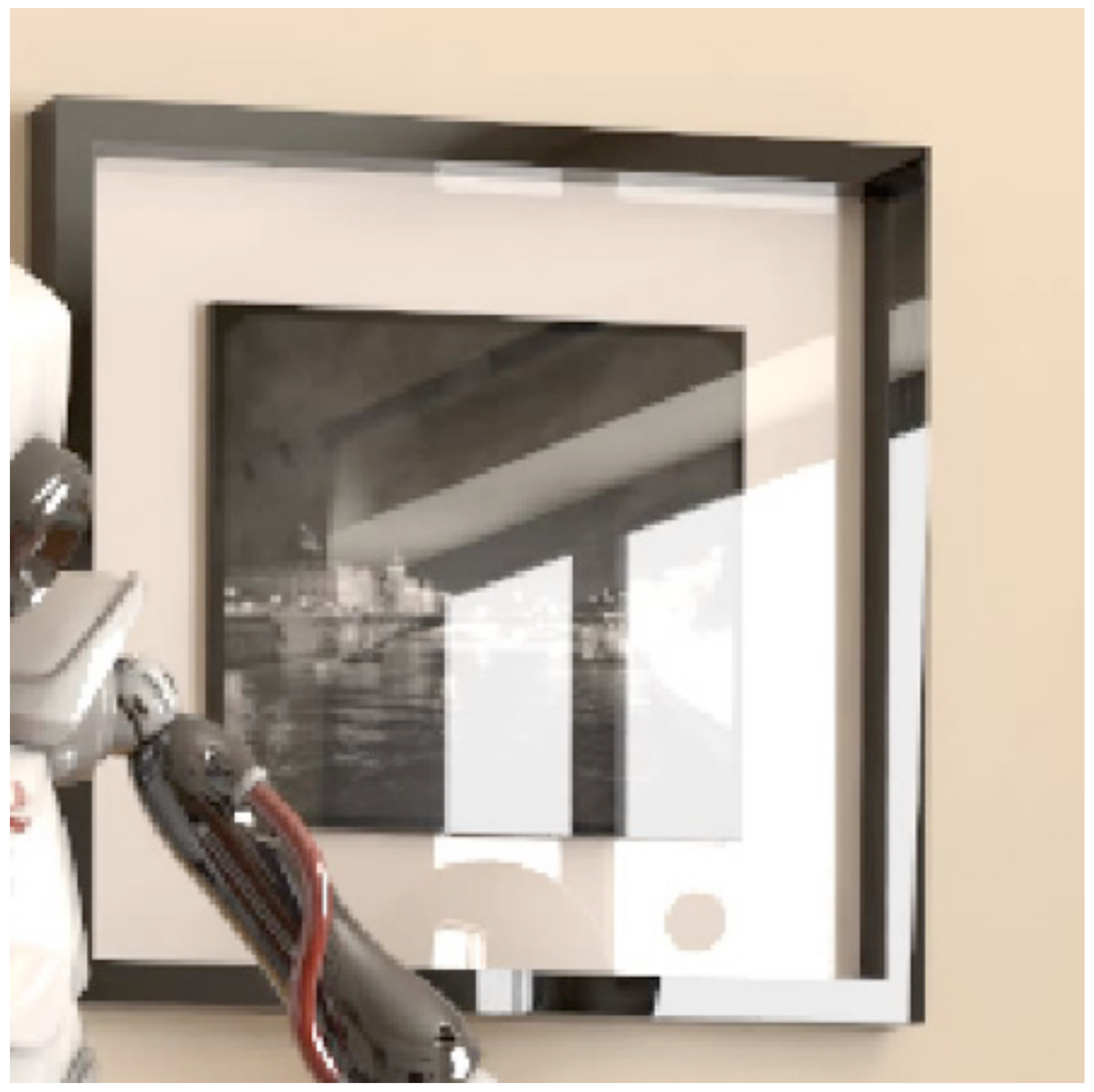
people want 4K renderings at high frame rate, and in stereo. The sheer amount of pixels that need to be rendered has become a significant computational burden, since these Monte Carlo techniques are just not fast enough.

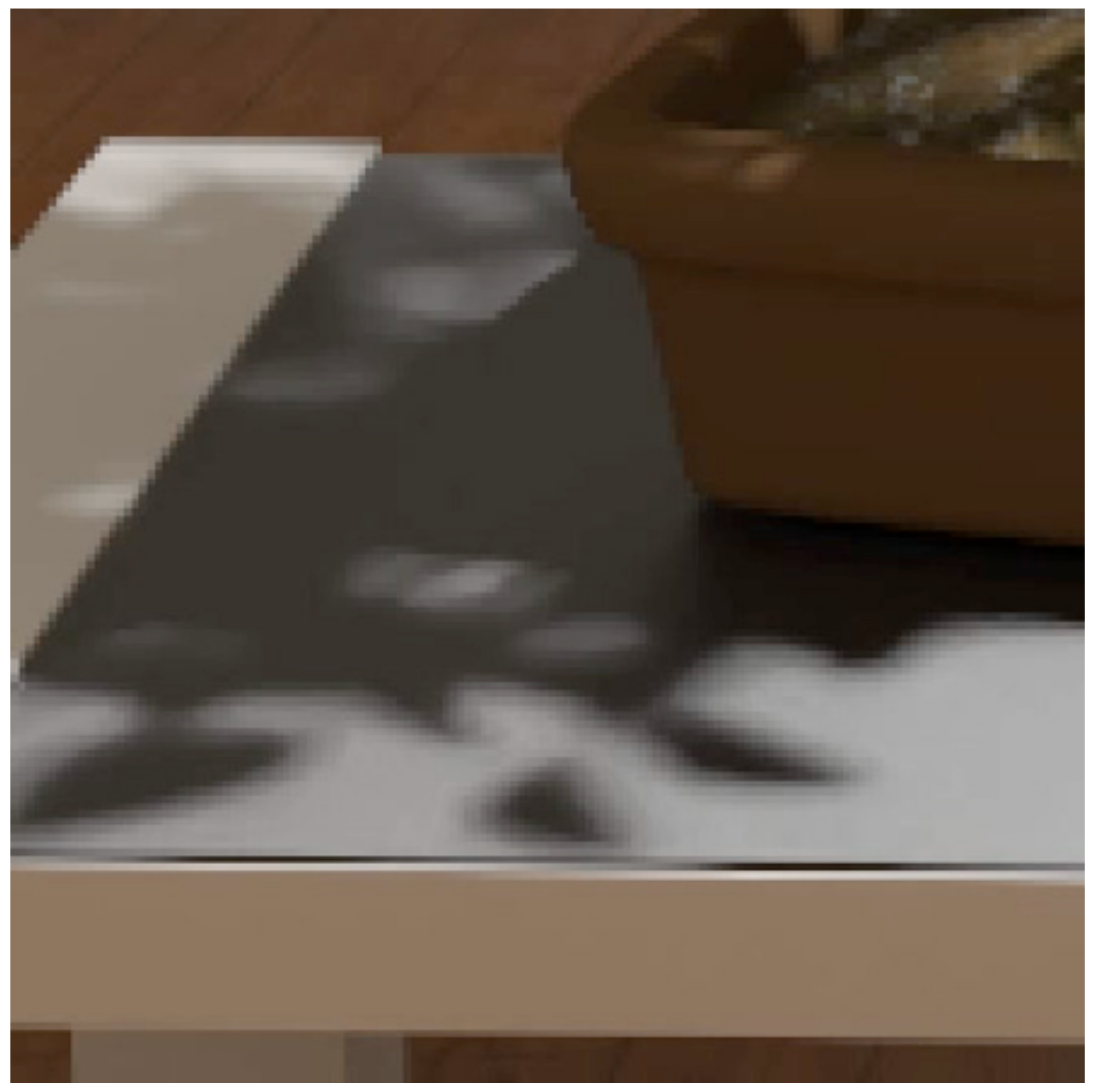
But there is so much coherence in space and time that we can in practice often render the data sparsely and recover the final image using image-based post-processing techniques.

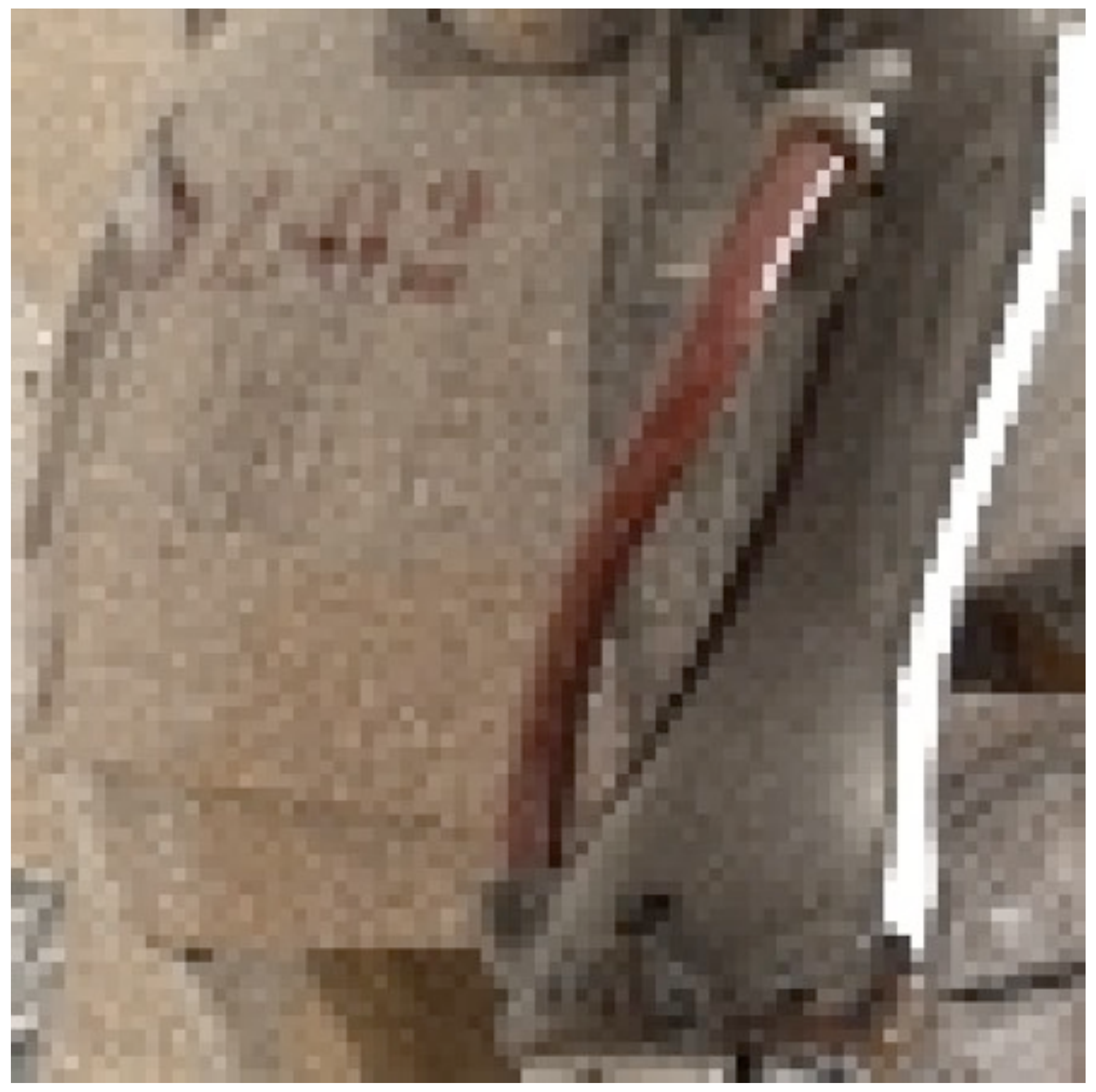
Here is an example output produced by our pipeline: given a noisy low-resolution and low-frame rate input, we create a smooth animation which has been spatio-temporally denoised and upsampled.

In order to produce this result using image-based post-processing, we must resolve a number of challenges.

Challenge: Ambiguity


Reflections


Shadows


Noise

Problem:
pixels are a composite of various components,
each with potentially distinct motion, spatial structure and noise level

The core issue is that image-based techniques rely on pixel data which is inherently ambiguous, since each pixel is a composite of various components with potentially distinct motion, spatial structure and noise level.

For instance, interpolation must take into account that reflections and shadows can have motion that is different from the underlying surfaces.

<click>
We solve the problem by decomposing the rendering into a set of disjoint components, each of which removes some ambiguity.

This turns out to make set of inherently ill-posed image processing problems surprisingly simple so that standard techniques, when applied on top of our decomposition, produce considerably better results.

Challenge: Ambiguity

Reflections

Shadows

Noise

Problem:
pixels are a composite of various components,
each with potentially distinct motion, spatial structure and noise level

Solution:
decomposition into disjoint components

The core issue is that image-based techniques rely on pixel data which is inherently ambiguous, since each pixel is a composite of various components with potentially distinct motion, spatial structure and noise level.

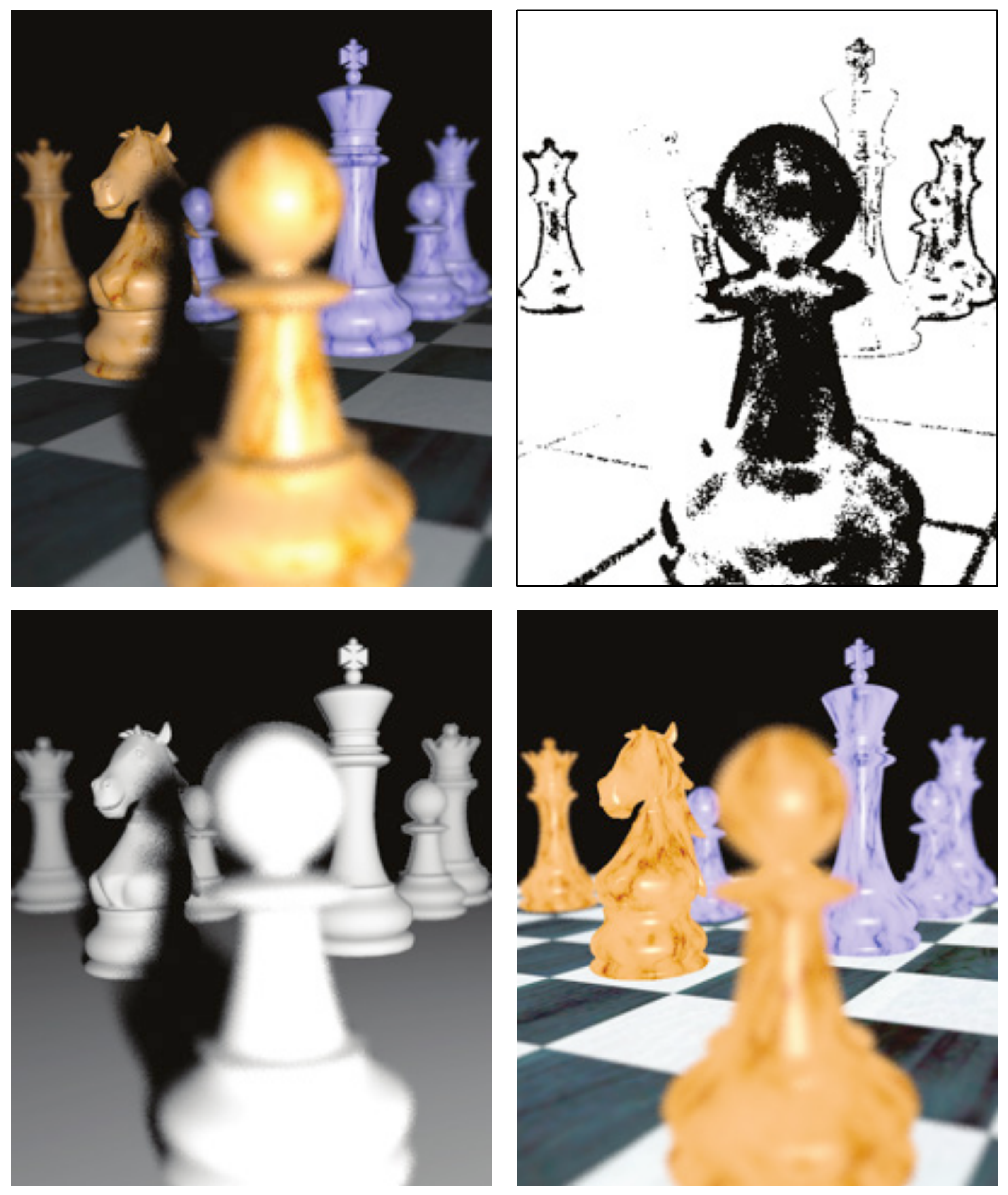
For instance, interpolation must take into account that reflections and shadows can have motion that is different from the underlying surfaces.

<click>
We solve the problem by decomposing the rendering into a set of disjoint components, each of which removes some ambiguity.

This turns out to make set of inherently ill-posed image processing problems surprisingly simple so that standard techniques, when applied on top of our decomposition, produce considerably better results.

Related Work

Irradiance factorization



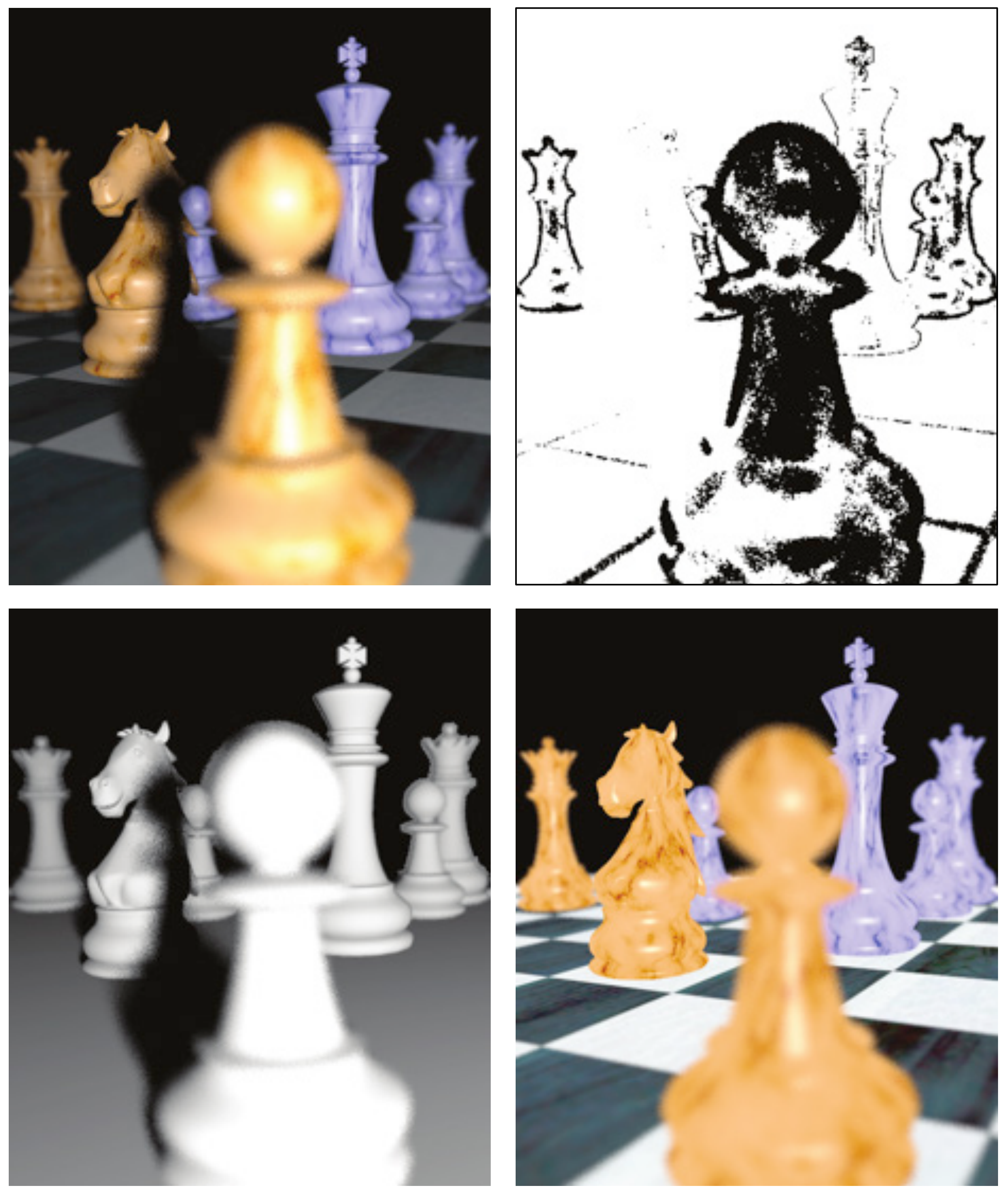
[Mehta et al. 2014]

Before describing our own method, I will briefly review some closely related works.

Separation of direct and indirect illumination, combined with irradiance factorization has been used in many works dating back to the seminal work on irradiance caching of Ward and colleagues to the recent denoising method of Mehta and colleagues.
<click>
Other works have focused on separate handling of specular interactions, such as this work by Lochman and colleagues which focused on novel view generation.
<click>
There is also a body of works that leverage output buffers encoding the scene information, such as depth and normals, to guide image-space filters. These are notably used for spatial upsampling and in denoising methods such as this one by Li and colleagues.

Related Work

Irradiance factorization



[Mehta et al. 2014]

Novel view synthesis for specular interactions

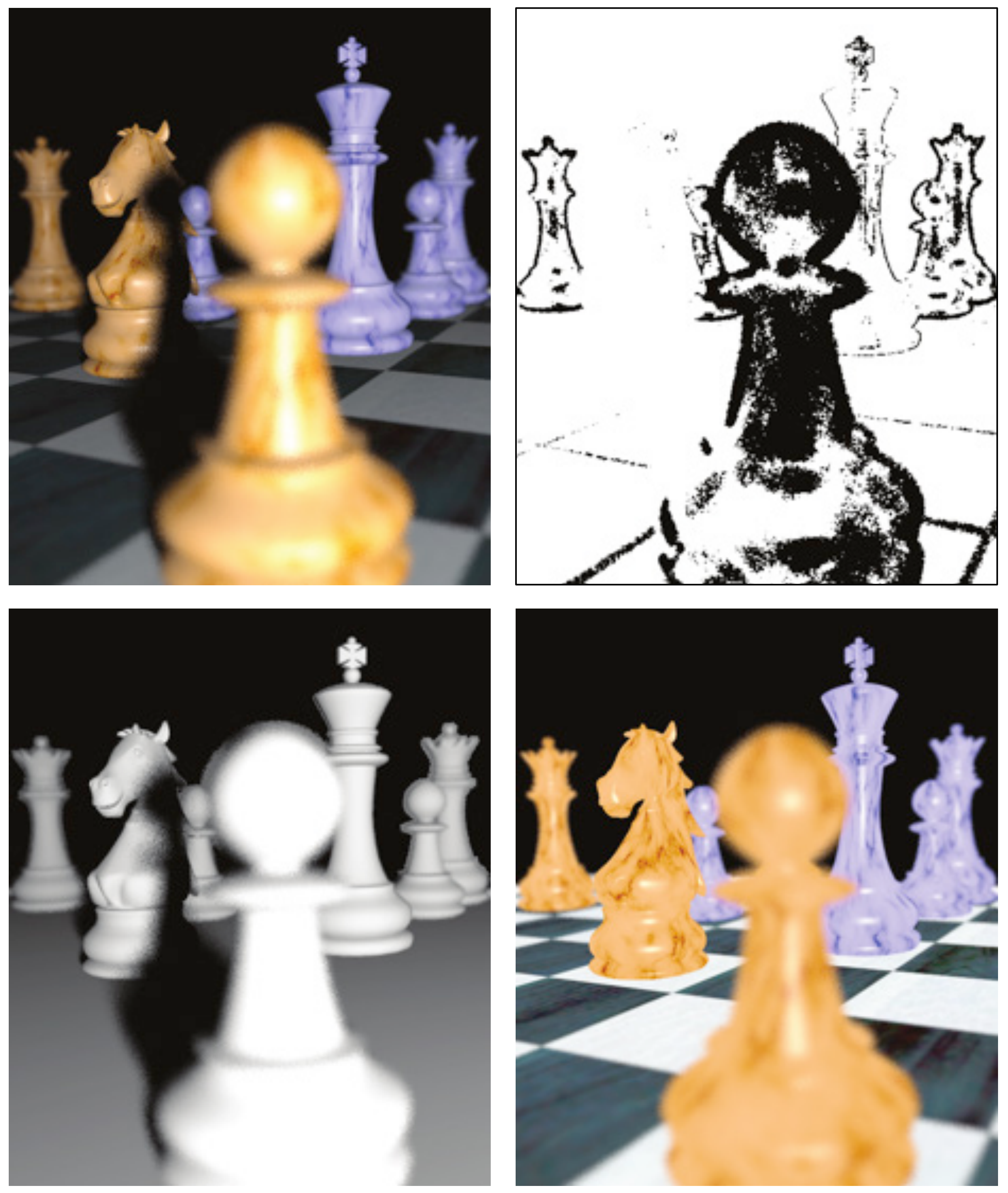
[Lochmann et al. 2014]

Before describing our own method, I will briefly review some closely related works.

Separation of direct and indirect illumination, combined with irradiance factorization has been used in many works dating back to the seminal work on irradiance caching of Ward and colleagues to the recent denoising method of Mehta and colleagues.
<click>
Other works have focused on separate handling of specular interactions, such as this work by Lochman and colleagues which focused on novel view generation.
<click>
There is also a body of works that leverage output buffers encoding the scene information, such as depth and normals, to guide image-space filters. These are notably used for spatial upsampling and in denoising methods such as this one by Li and colleagues.

Related Work

Irradiance factorization

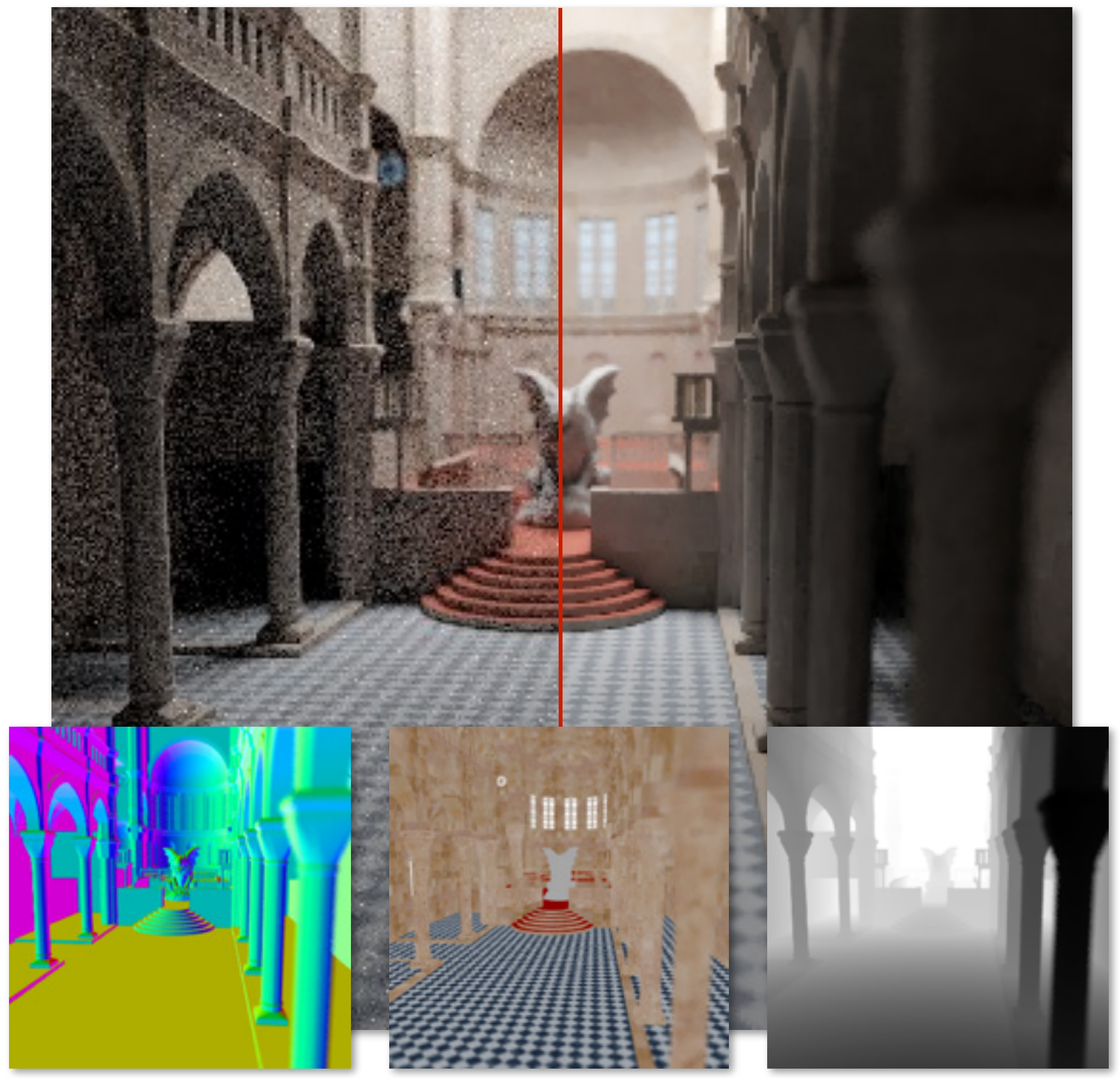


[Mehta et al. 2014]

Novel view synthesis for specular interactions

[Lochmann et al. 2014]

Joint filtering using feature buffers



[Li et al. 2012]

Before describing our own method, I will briefly review some closely related works.

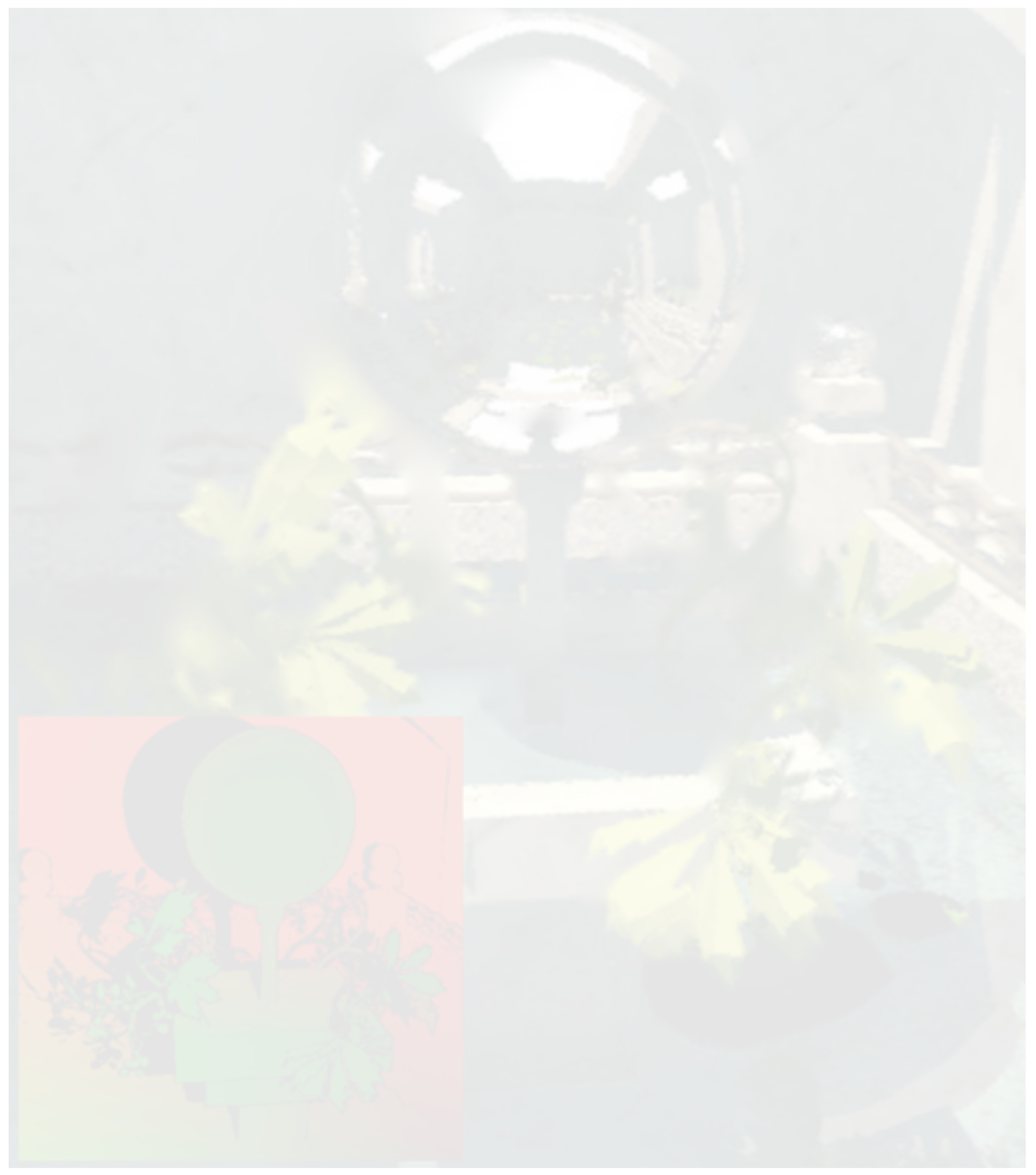
Separation of direct and indirect illumination, combined with irradiance factorization has been used in many works dating back to the seminal work on irradiance caching of Ward and colleagues to the recent denoising method of Mehta and colleagues.
<click>
Other works have focused on separate handling of specular interactions, such as this work by Lochman and colleagues which focused on novel view generation.
<click>
There is also a body of works that leverage output buffers encoding the scene information, such as depth and normals, to guide image-space filters. These are notably used for spatial upsampling and in denoising methods such as this one by Li and colleagues.

Related Work

Irradiance factorization

[Mehta et al. 2014]

Novel view synthesis for specular interactions



[Lochmann et al. 2014]

Joint filtering using feature buffers

[Li et al. 2012]

Our own work consolidates these previous decompositions in a cohesive framework, with some key modifications.
<click>
In particular we introduce a novel effective irradiance factorization that can handle non-Lambertian materials as well as depth-of-field and motion blur,
<click>
we also propose a method to accurately compute motion vectors for reflections and refractions,
<click>
and lastly we use a richer set of feature buffers specific to each component of our decomposition.

But before going in more detail on our contributions, I will briefly describe the decomposition used in our pipeline.

Related Work

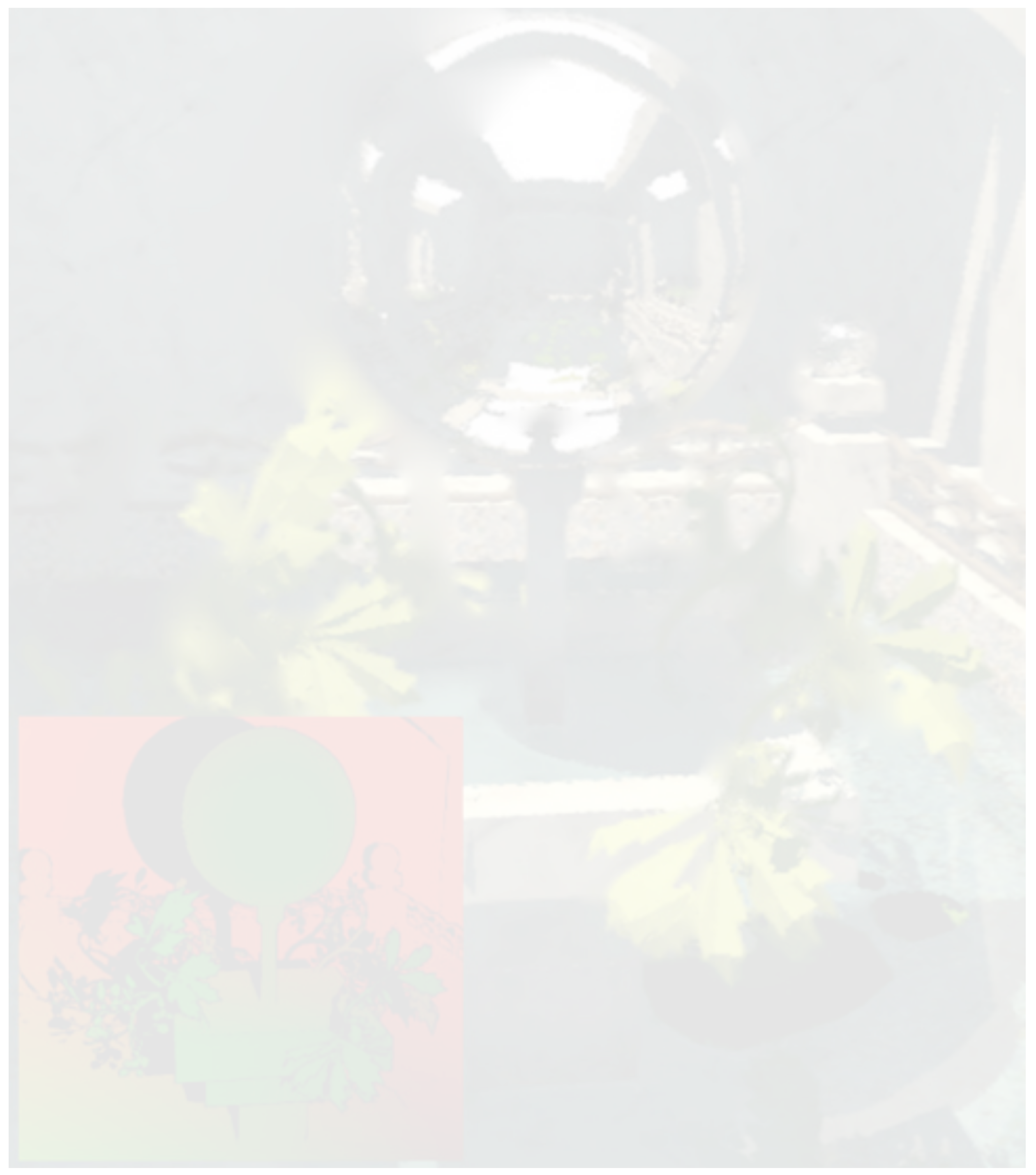
Our contributions:

Irradiance factorization

Novel view synthesis for specular interactions

Joint filtering using feature buffers

[Mehta et al. 2014]



[Lochmann et al. 2014]

[Li et al. 2012]

Our own work consolidates these previous decompositions in a cohesive framework, with some key modifications.
<click>
In particular we introduce a novel effective irradiance factorization that can handle non-Lambertian materials as well as depth-of-field and motion blur,
<click>
we also propose a method to accurately compute motion vectors for reflections and refractions,
<click>
and lastly we use a richer set of feature buffers specific to each component of our decomposition.

But before going in more detail on our contributions, I will briefly describe the decomposition used in our pipeline.

Related Work

Our contributions:

- Effective irradiance factorization

Irradiance factorization

[Mehta et al. 2014]

Novel view synthesis for specular interactions

[Lochmann et al. 2014]

Joint filtering using feature buffers

[Li et al. 2012]

Our own work consolidates these previous decompositions in a cohesive framework, with some key modifications.
<click>
In particular we introduce a novel effective irradiance factorization that can handle non-Lambertian materials as well as depth-of-field and motion blur,
<click>
we also propose a method to accurately compute motion vectors for reflections and refractions,
<click>
and lastly we use a richer set of feature buffers specific to each component of our decomposition.

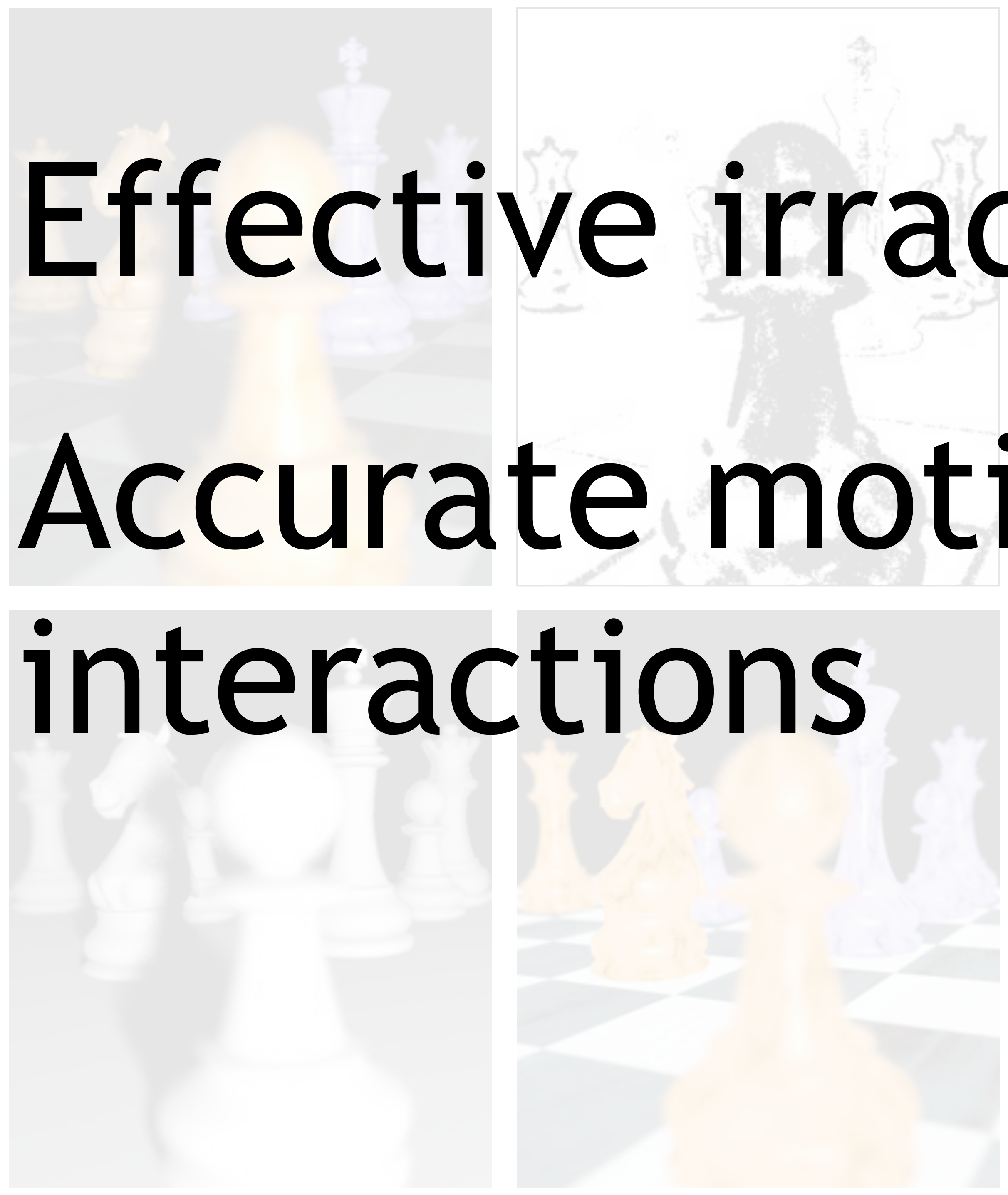
But before going in more detail on our contributions, I will briefly describe the decomposition used in our pipeline.

Related Work

Our contributions:

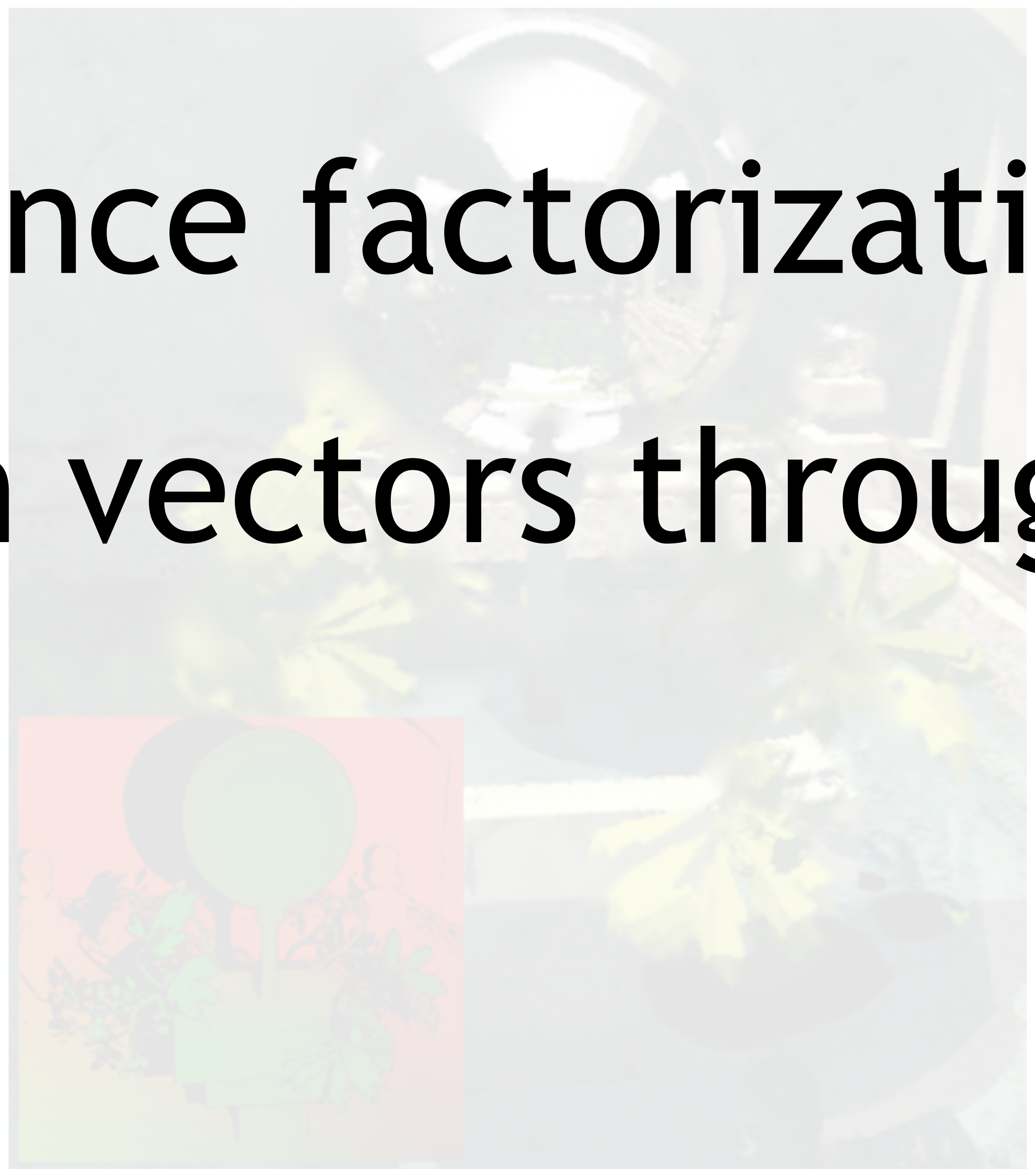
- Effective irradiance factorization
- Accurate motion vectors through specular interactions

Irradiance factorization



[Mehta et al. 2014]

Novel view synthesis for specular interactions



[Lochmann et al. 2014]

Joint filtering using feature buffers

[Li et al. 2012]

Our own work consolidates these previous decompositions in a cohesive framework, with some key modifications.
<click>
In particular we introduce a novel effective irradiance factorization that can handle non-Lambertian materials as well as depth-of-field and motion blur,
<click>
we also propose a method to accurately compute motion vectors for reflections and refractions,
<click>
and lastly we use a richer set of feature buffers specific to each component of our decomposition.

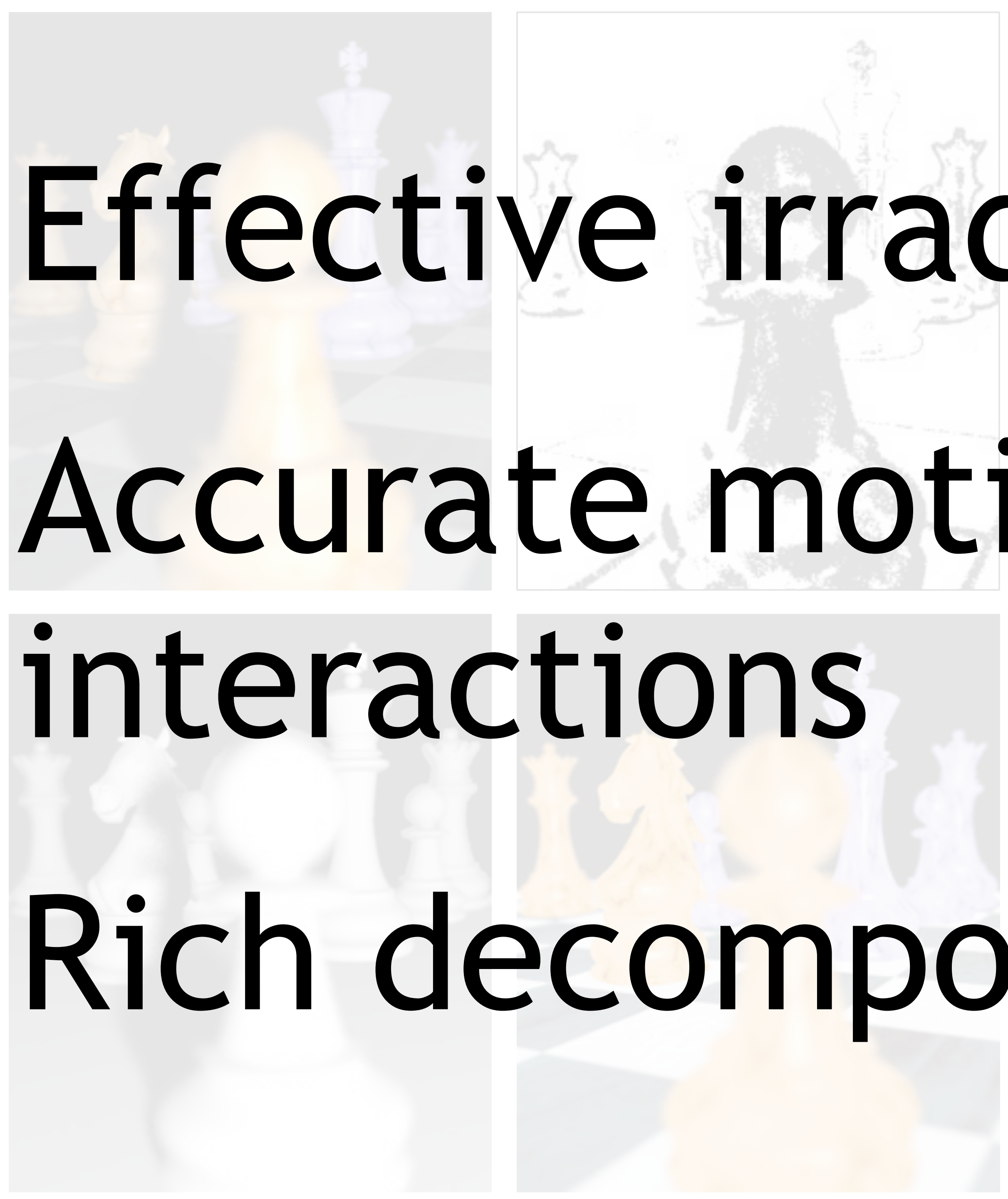
But before going in more detail on our contributions, I will briefly describe the decomposition used in our pipeline.

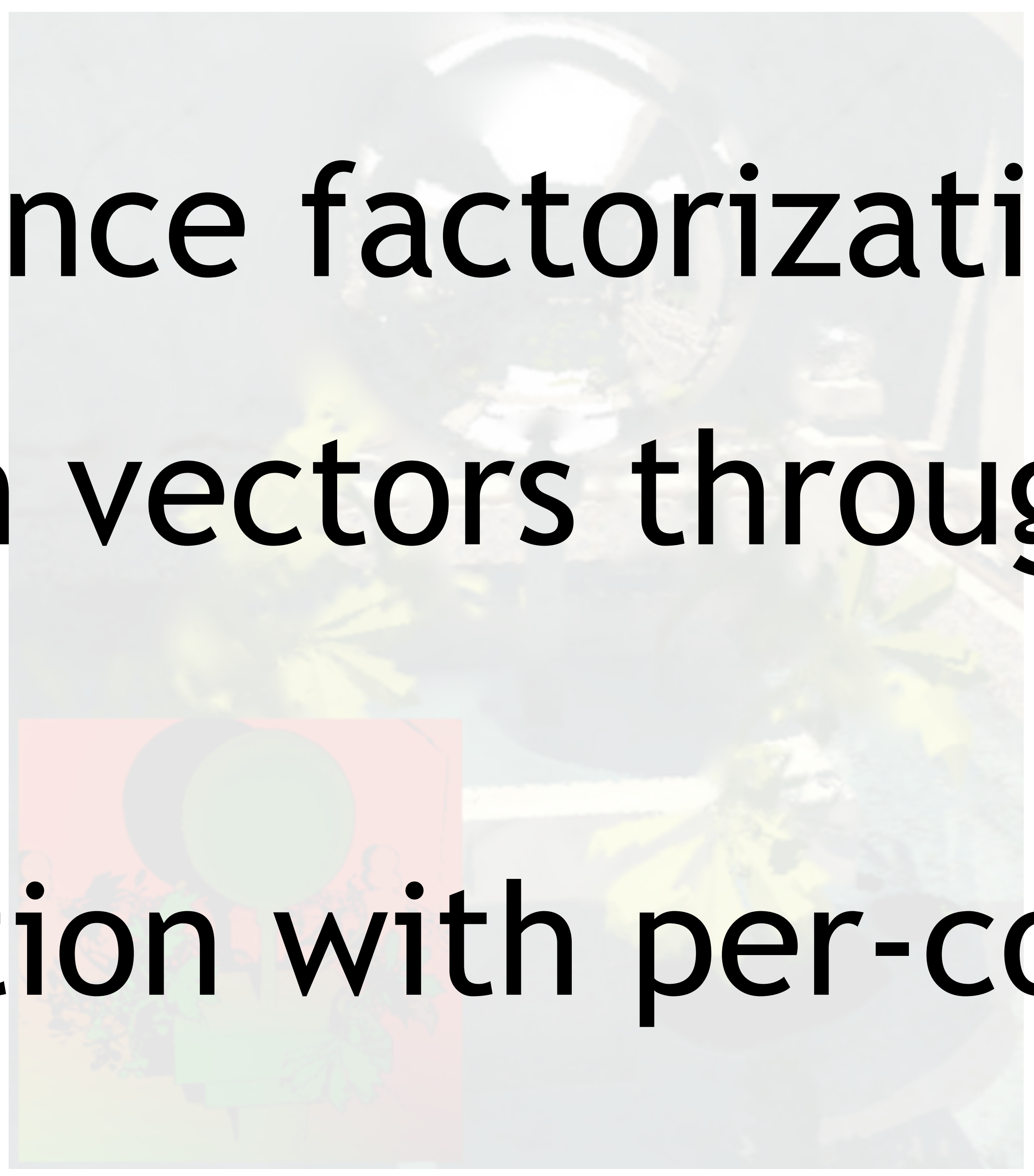
Related Work

Our contributions:

Irradiance factorization | Novel view synthesis for specular interactions | Joint filtering using feature buffers

- Effective irradiance factorization
- Accurate motion vectors through specular interactions
- Rich decomposition with per-component features


[Mehta et al. 2014]

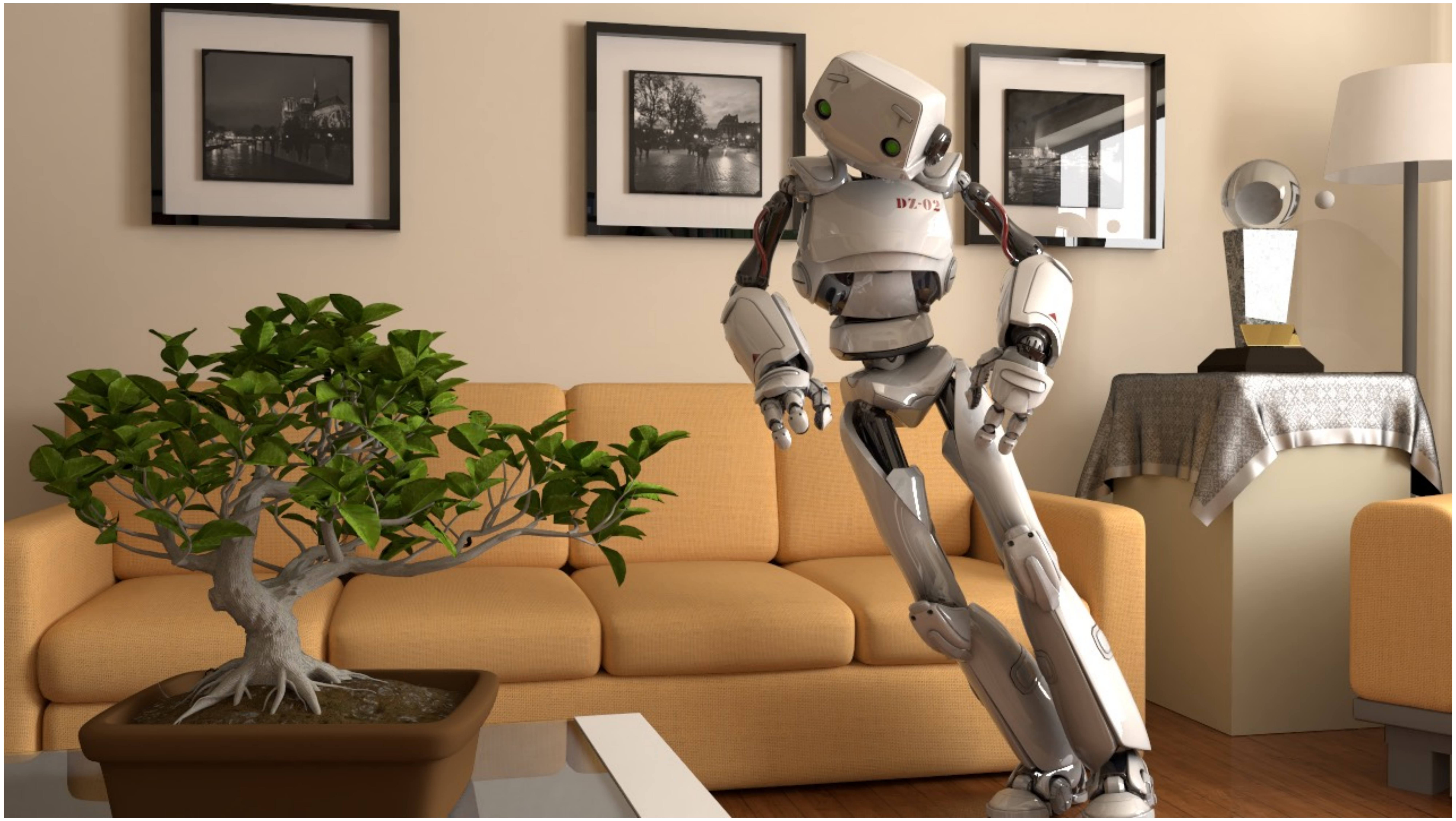

[Lochmann et al. 2014]

[Li et al. 2012]

Our own work consolidates these previous decompositions in a cohesive framework, with some key modifications.
<click>
In particular we introduce a novel effective irradiance factorization that can handle non-Lambertian materials as well as depth-of-field and motion blur,
<click>
we also propose a method to accurately compute motion vectors for reflections and refractions,
<click>
and lastly we use a richer set of feature buffers specific to each component of our decomposition.

But before going in more detail on our contributions, I will briefly describe the decomposition used in our pipeline.

Path-Space Decomposition



Our system supports arbitrary decompositions, and here I'll just show you a specific one, using our robot scene as an example.

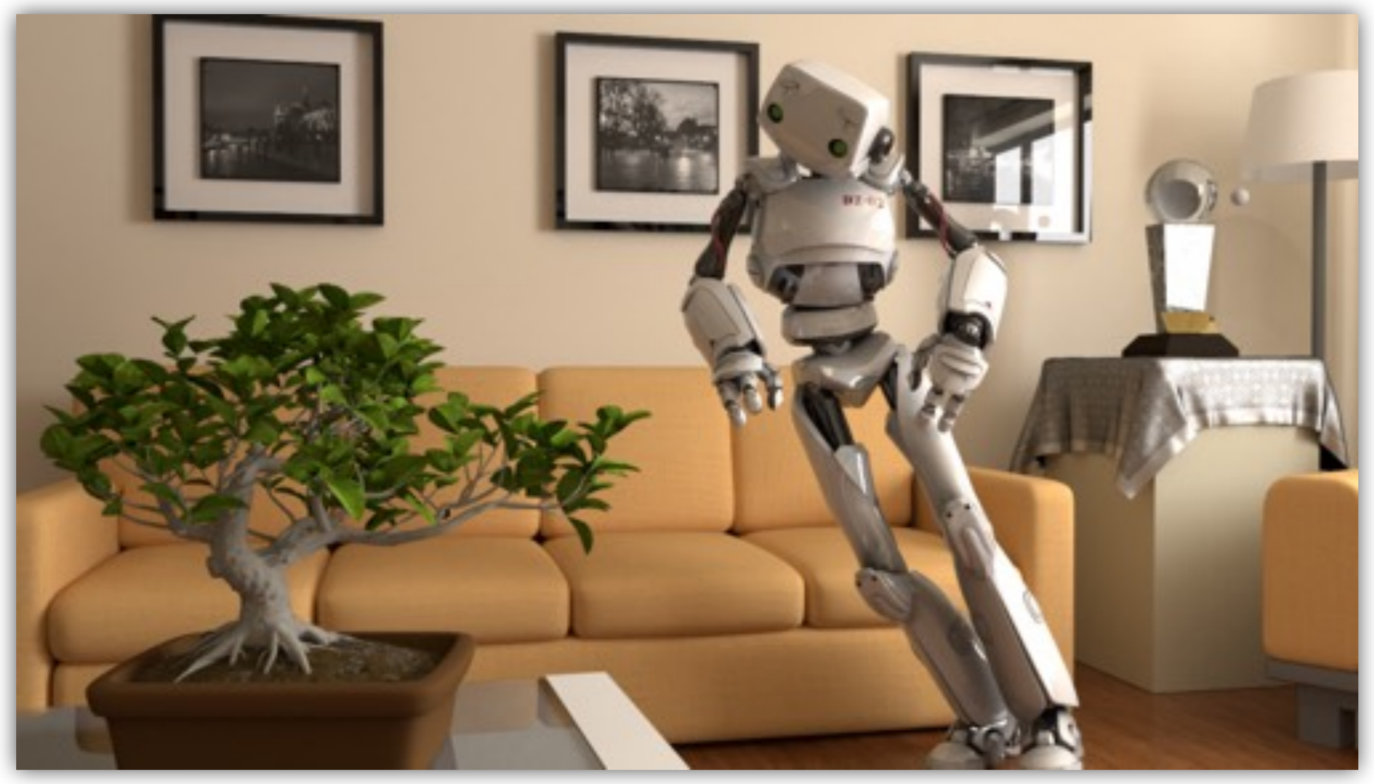
Color

=

Path specification using regular expressions

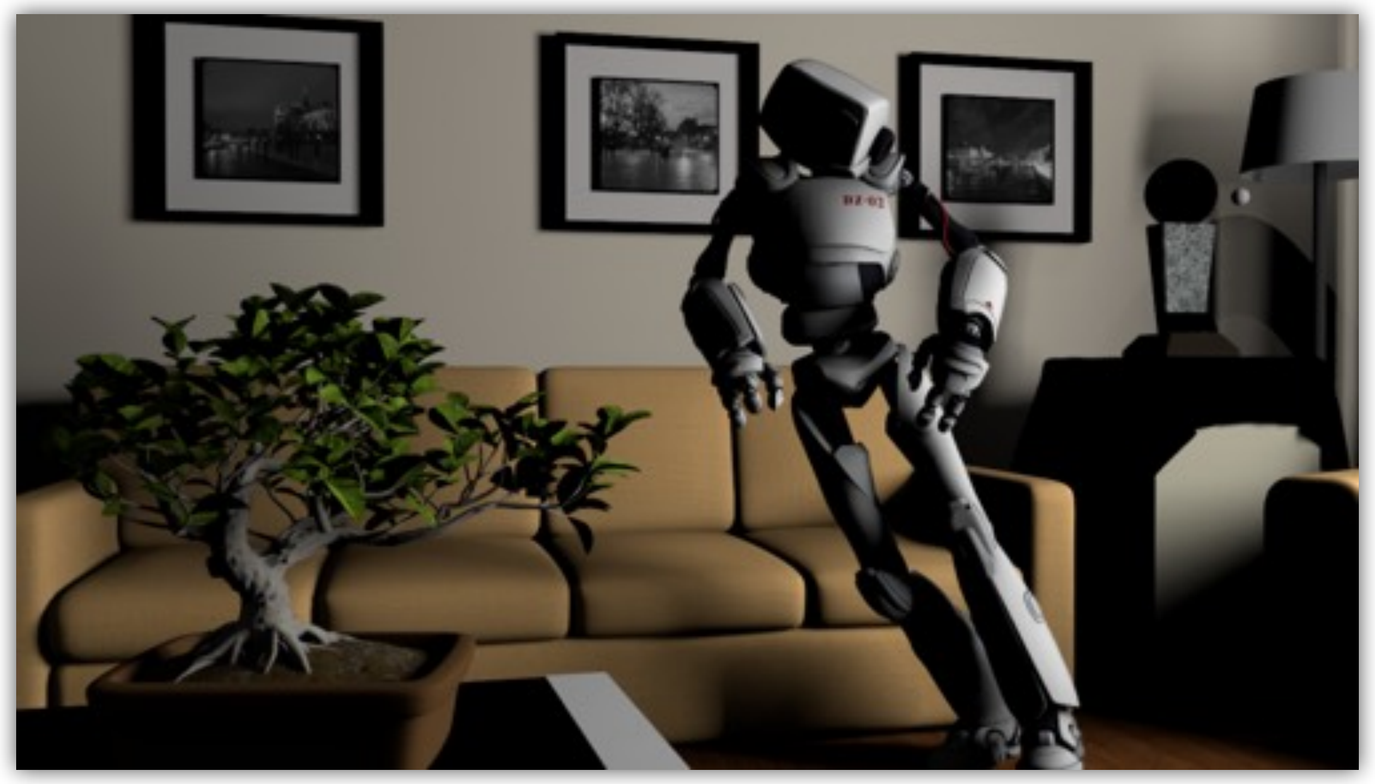
Our goal is to split the output of a path-tracer in a set of disjoint components and we use regular expressions to define to which component a specific path contributes.

Color



=

Diffuse Direct – **ED**



+

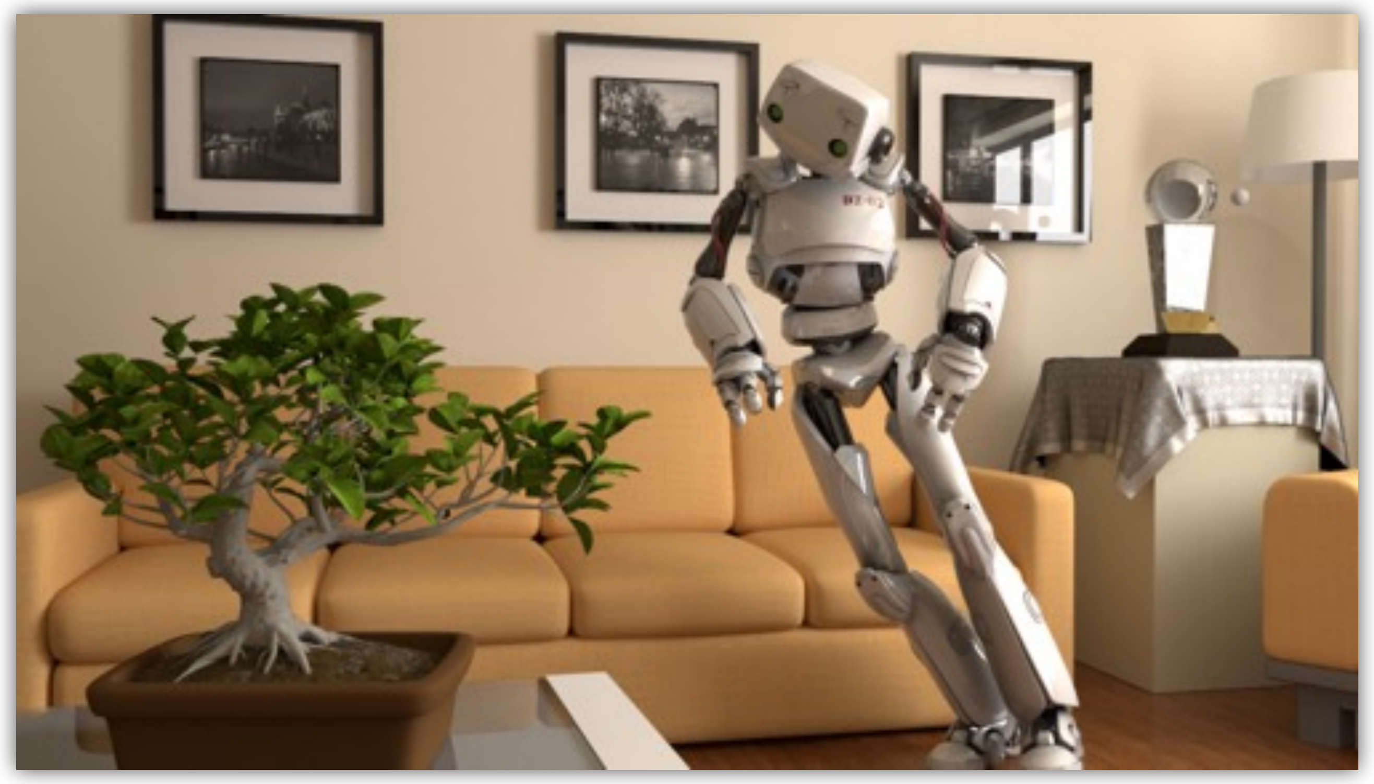
Path specification using regular expressions

Diffuse direct: ED (Eye – Diffuse)

Disney Research igl ETH zürich Walt Disney Animation Studios

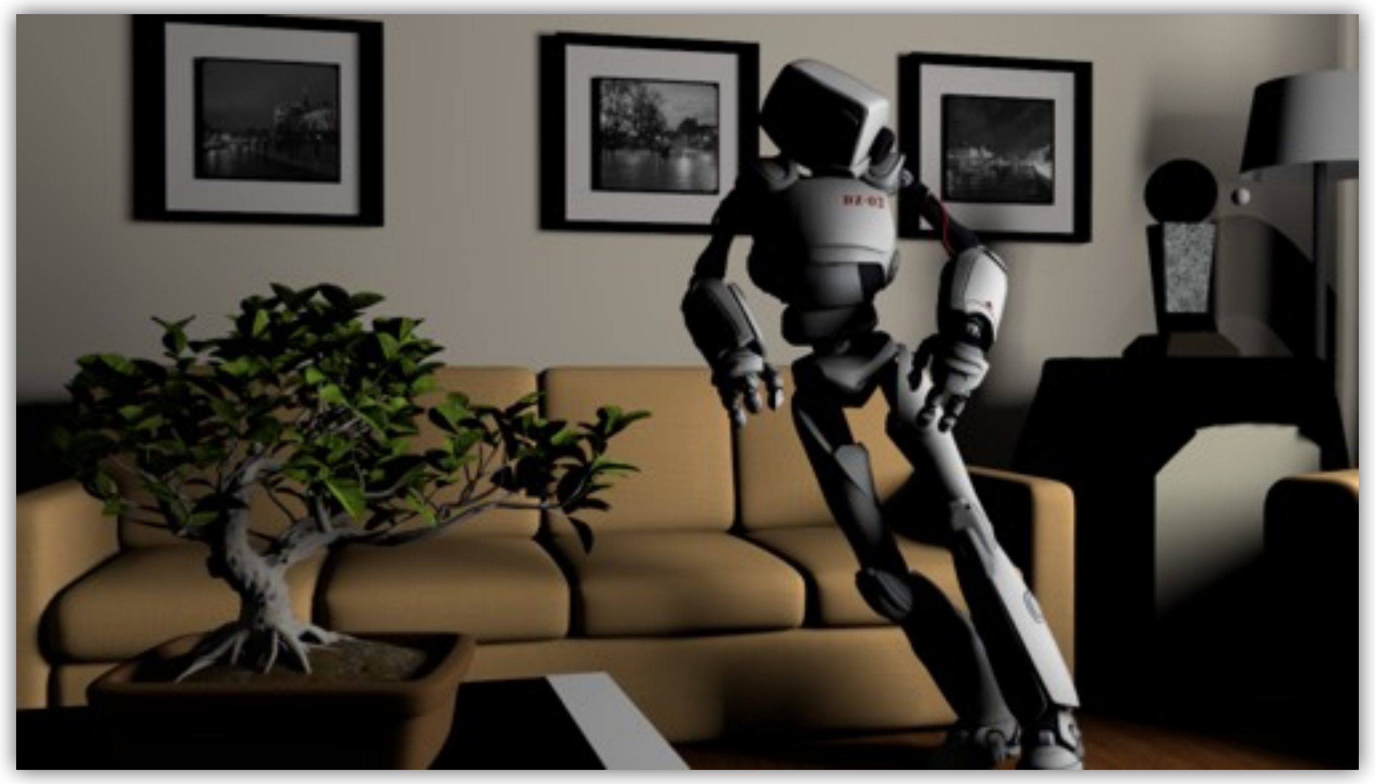
We start by separating out the direct diffuse component, that is, diffuse surfaces that are directly visible from the eye or camera. Using Heckbert notation, those are called ED paths.

Color



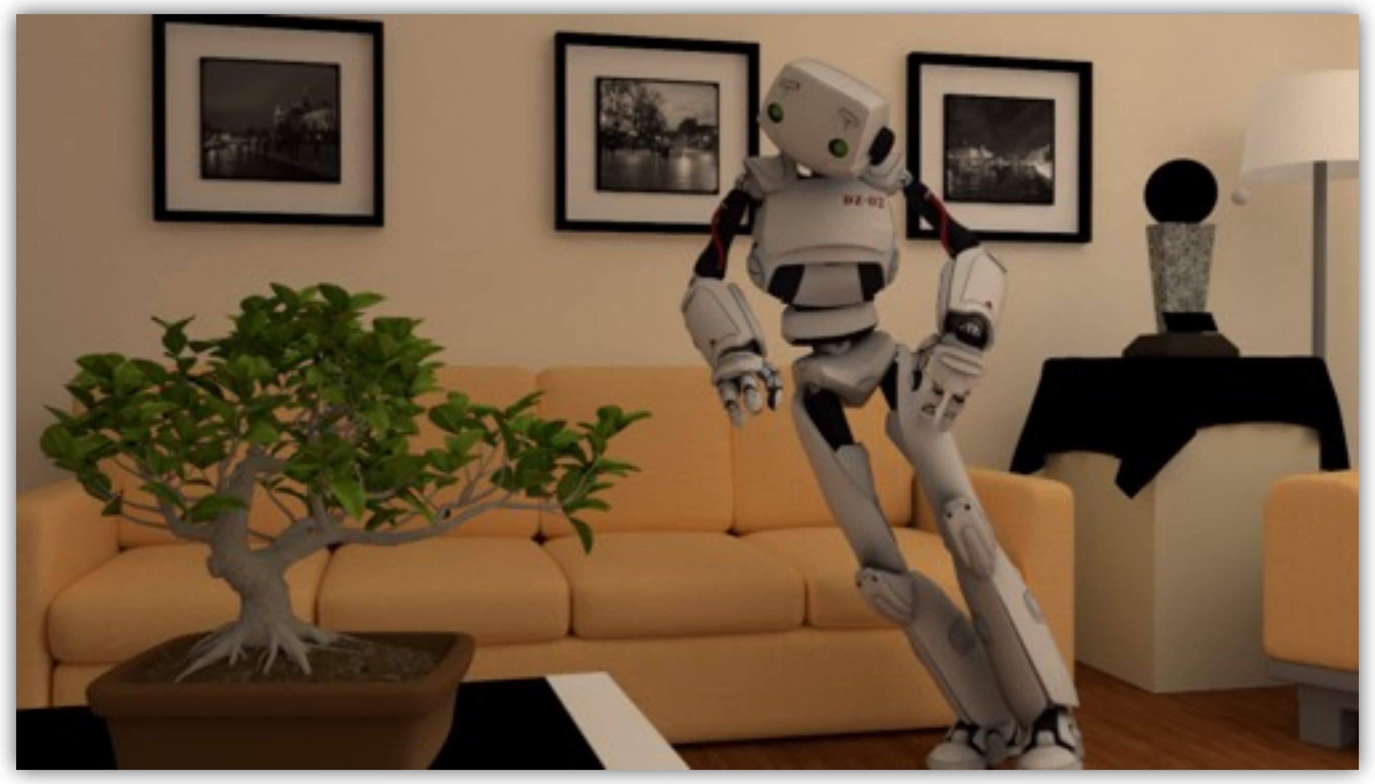
=

Diffuse Direct – **ED**



+

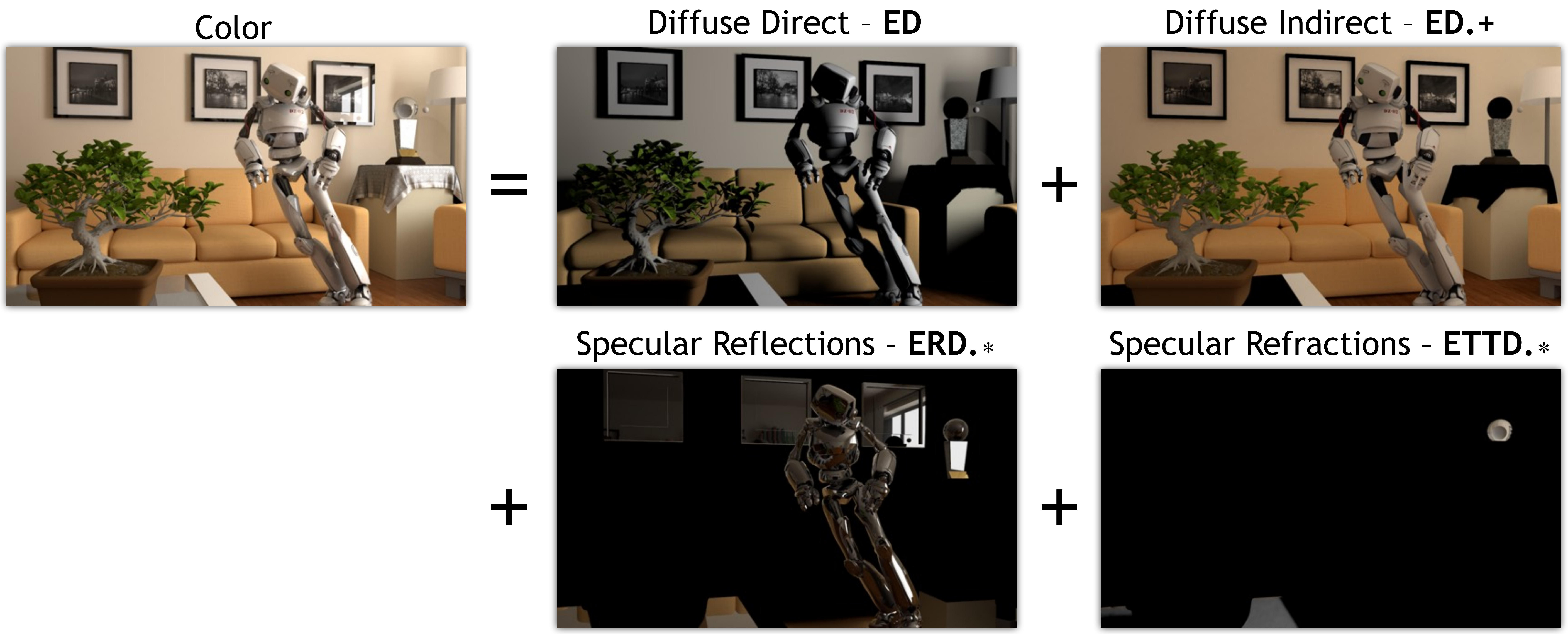
Diffuse Indirect – **ED.+**



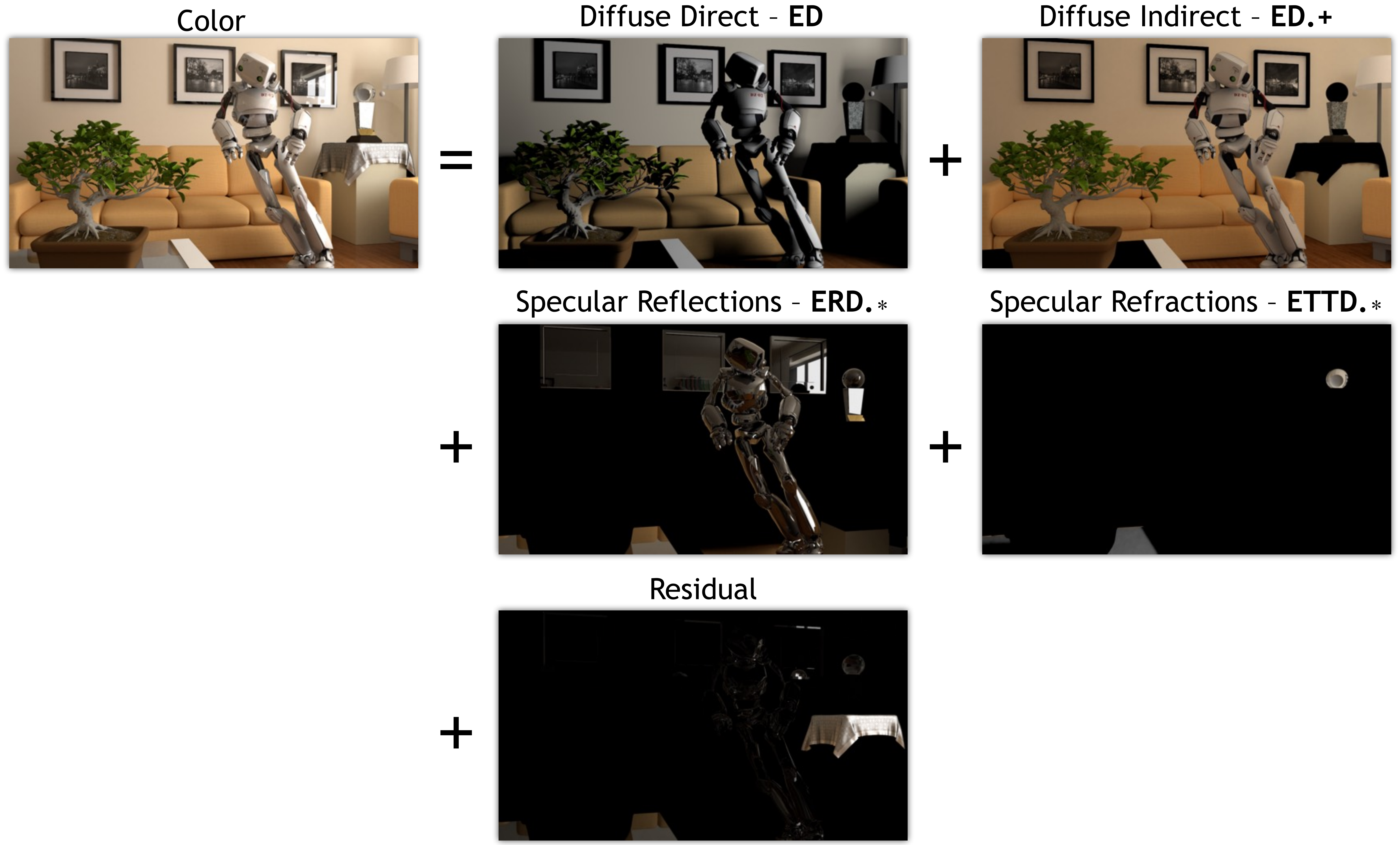
Path specification using regular expressions

- Diffuse direct: ED (Eye – Diffuse)
- Diffuse indirect: ED.+ (Eye – Diffuse, plus at least one event)

We can similarly define the diffuse indirect component…


Color
=
Diffuse Direct – **ED**
+
Diffuse Indirect – **ED.+**
+
Specular Reflections – **ERD.***
+
Specular Refractions – **ETTD.***

As well as the specular reflection and specular refraction components



And any type of path not captured by one of the previous components is simply redirected to the residual component.

In practice, all these components will be extracted in a single rendering pass, using a path tracing integrator with a Final State Machine (FSM) to track what component should be updated.

Depending on the scene, we may want to extract additional components as we will see later.

Per-component Features

We will also enrich our decomposition with a set of feature buffers specific to each component.

These are the features we capture for the diffuse direct component.

<click>
First, we extract information from the visible geometry, including the normal, shape index and surface reflectance.

<click>
We then compute what we call the effective irradiance, which I will explain in just a moment.

<click>
Finally, we have two motion vector buffers, one that encodes the motion of the geometry in the scene, and one that encodes the motion of the irradiance alone.

Per-component Features

Color

Diffuse Direct

Monday, June 29, 15

We will also enrich our decomposition with a set of feature buffers specific to each component.

These are the features we capture for the diffuse direct component.

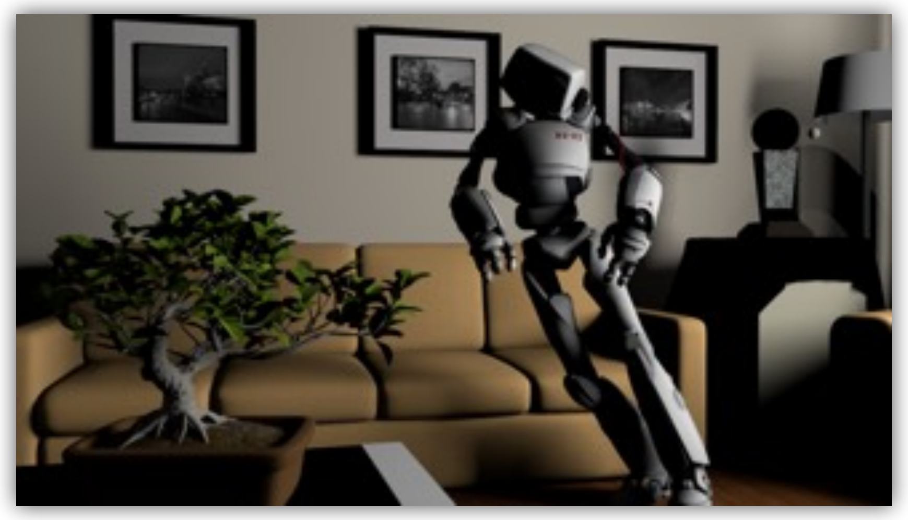
<click>
First, we extract information from the visible geometry, including the normal, shape index and surface reflectance.

<click>
We then compute what we call the effective irradiance, which I will explain in just a moment.

<click>
Finally, we have two motion vector buffers, one that encodes the motion of the geometry in the scene, and one that encodes the motion of the irradiance alone.

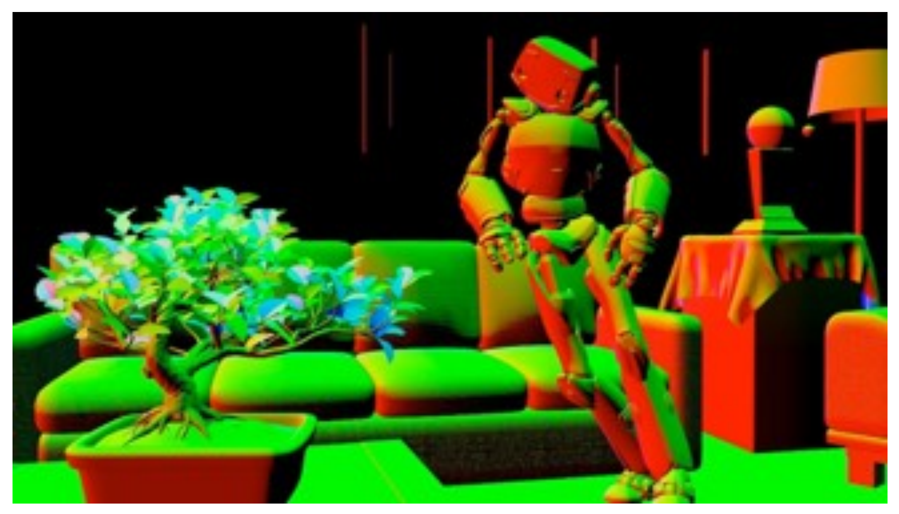
Per-component Features

Color

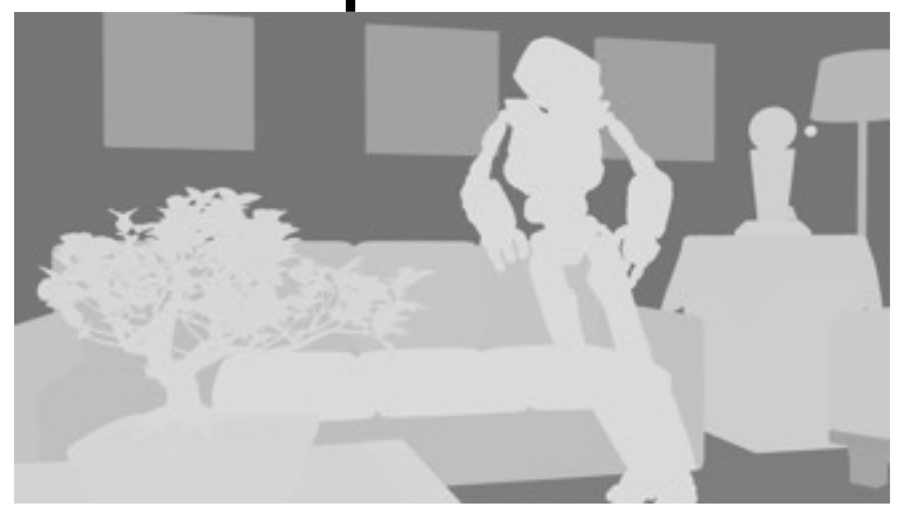


Diffuse Direct

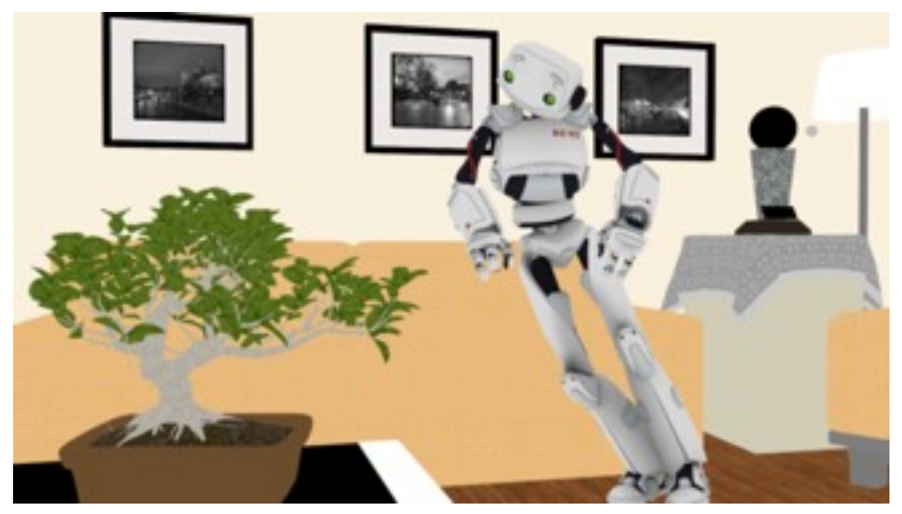
Normal



Shape Index



Reflectance



Monday, June 29, 15

We will also enrich our decomposition with a set of feature buffers specific to each component.

These are the features we capture for the diffuse direct component.

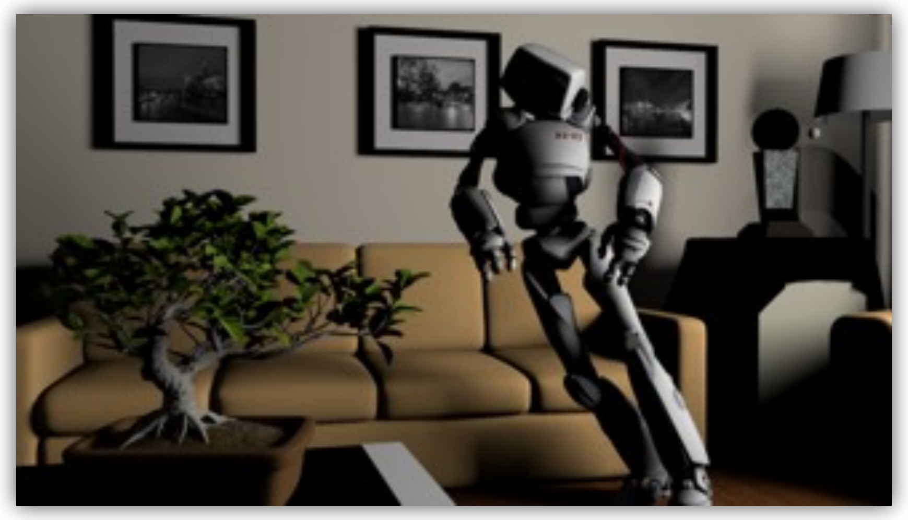
<click>
First, we extract information from the visible geometry, including the normal, shape index and surface reflectance.

<click>
We then compute what we call the effective irradiance, which I will explain in just a moment.

<click>
Finally, we have two motion vector buffers, one that encodes the motion of the geometry in the scene, and one that encodes the motion of the irradiance alone.

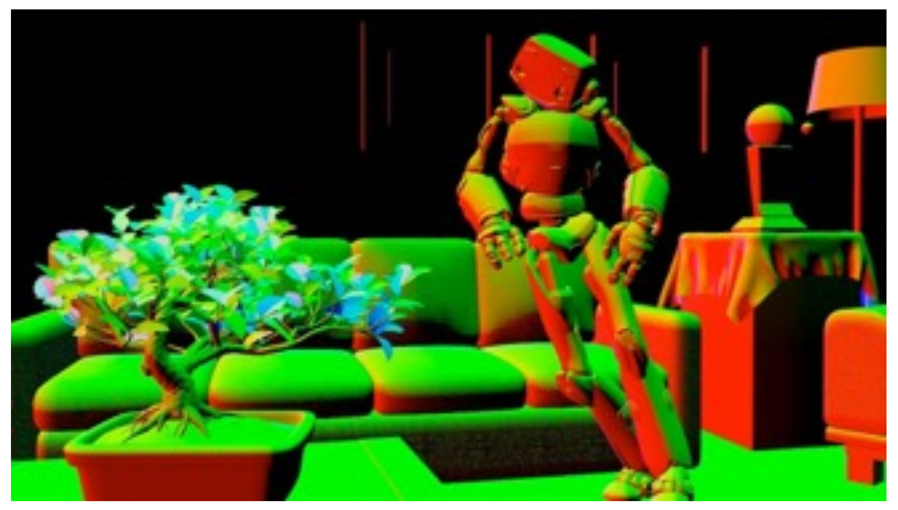
Per-component Features

Color

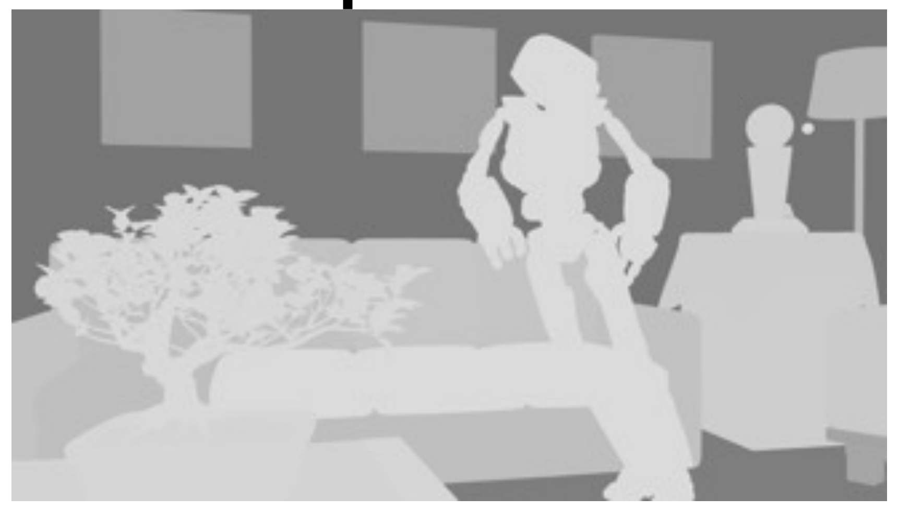


Diffuse Direct

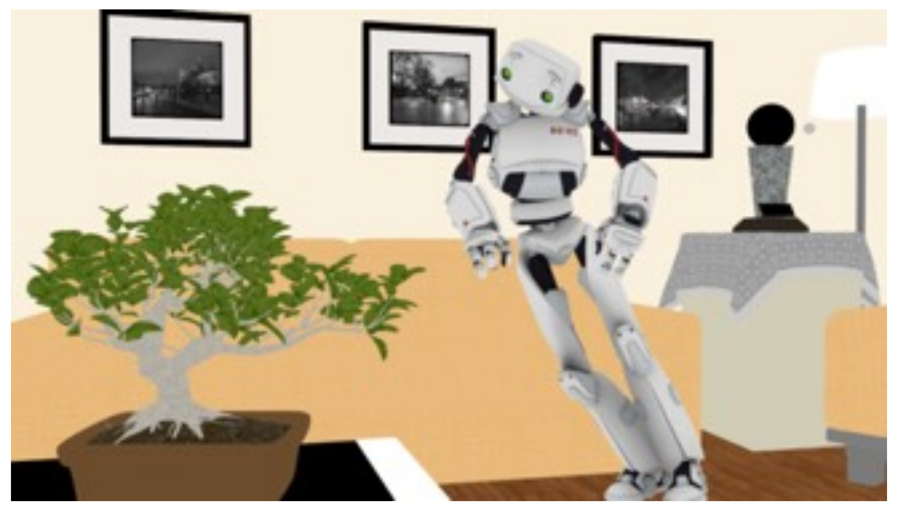
Normal



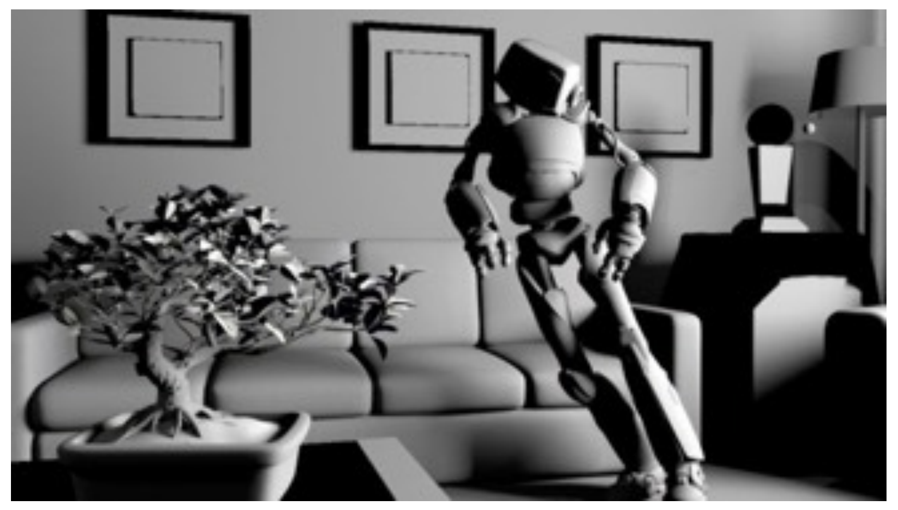
Shape Index



Reflectance



Eff. Irradiance



We will also enrich our decomposition with a set of feature buffers specific to each component.

These are the features we capture for the diffuse direct component.

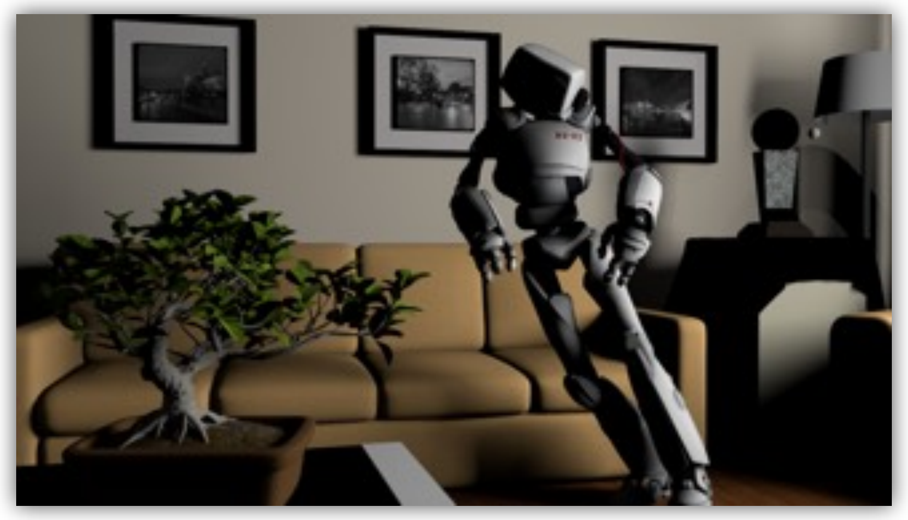
<click>
First, we extract information from the visible geometry, including the normal, shape index and surface reflectance.

<click>
We then compute what we call the effective irradiance, which I will explain in just a moment.

<click>
Finally, we have two motion vector buffers, one that encodes the motion of the geometry in the scene, and one that encodes the motion of the irradiance alone.

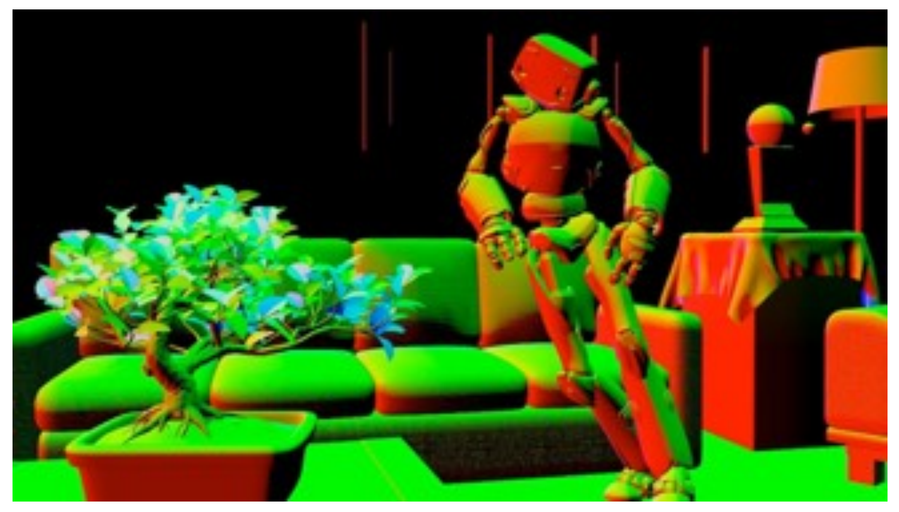
Per-component Features

Color

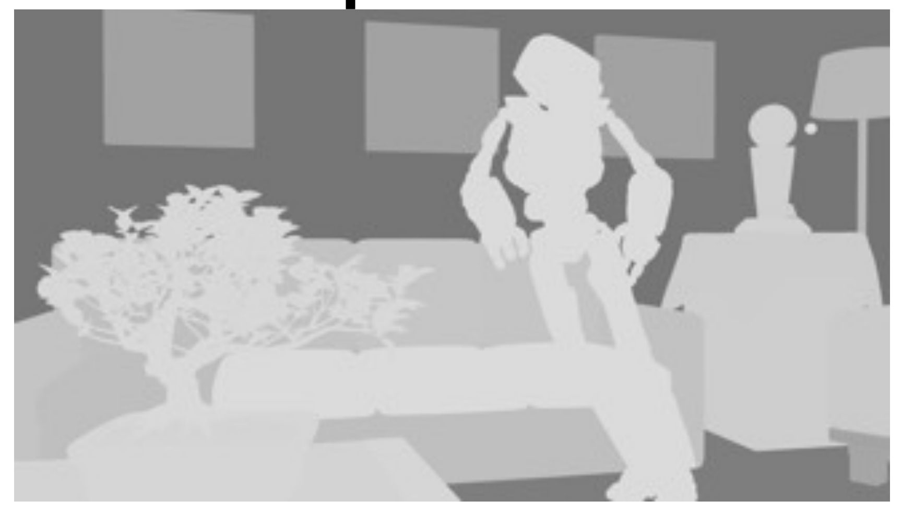


Diffuse Direct

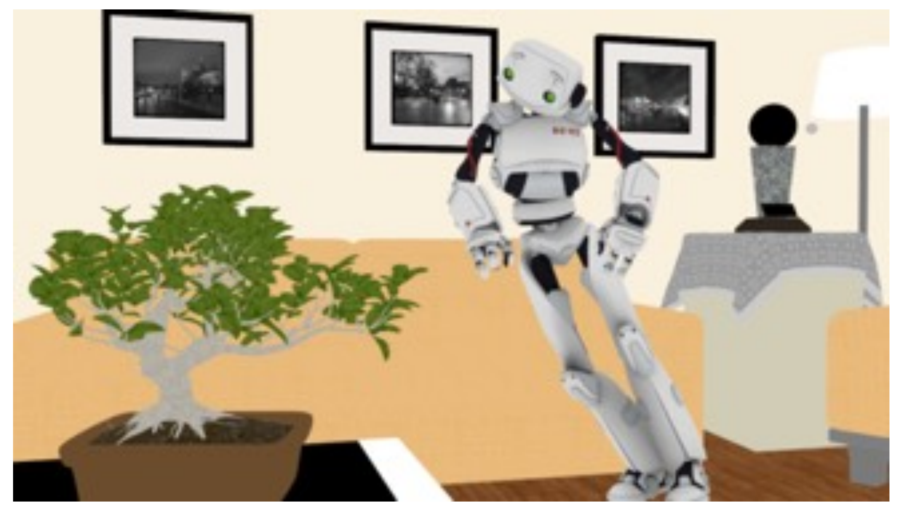
Normal



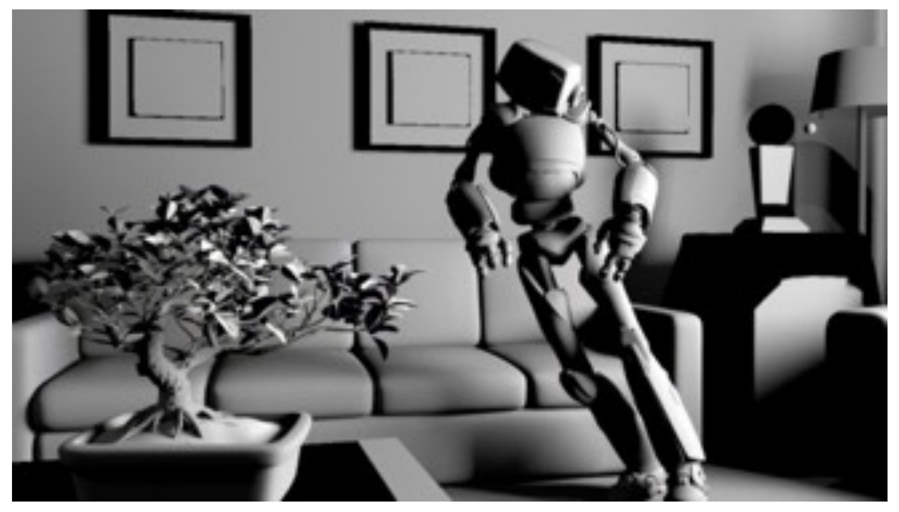
Shape Index



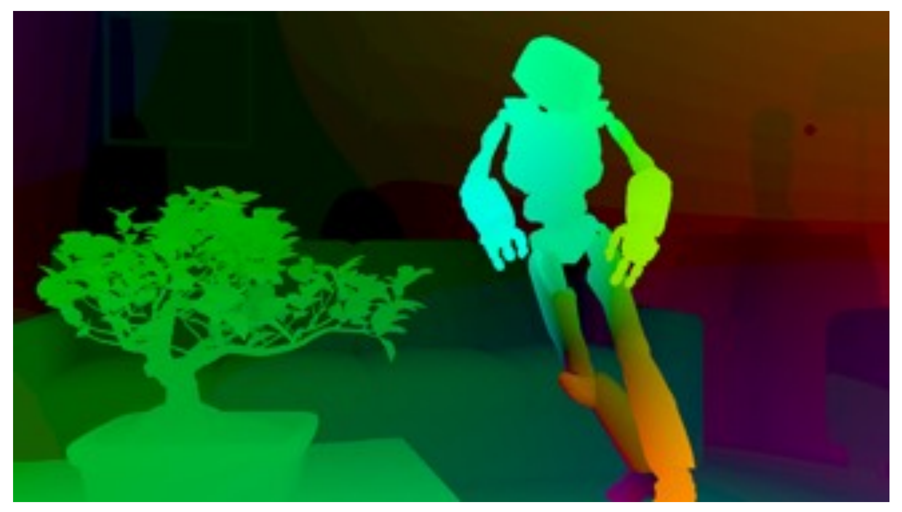
Reflectance



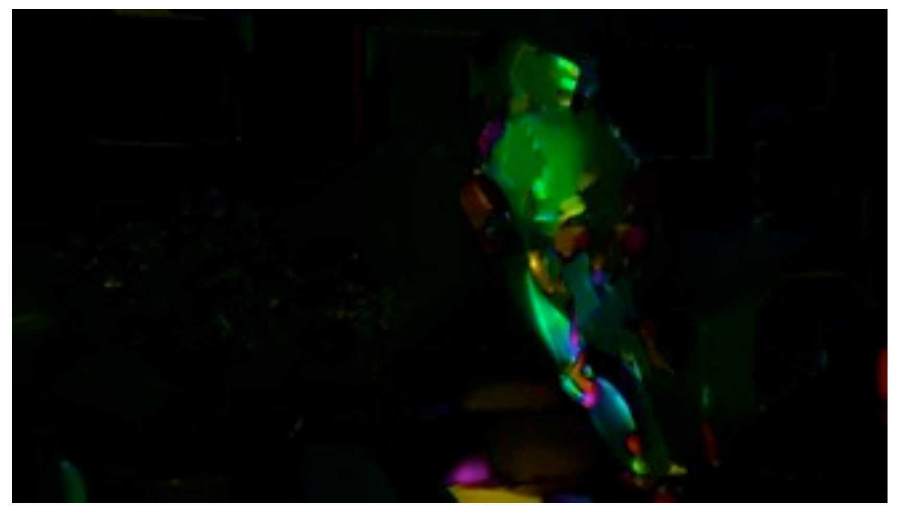
Eff. Irradiance



Geom. Motion



Irradiance Motion



Monday, June 29, 15

We will also enrich our decomposition with a set of feature buffers specific to each component.

These are the features we capture for the diffuse direct component.

<click>
First, we extract information from the visible geometry, including the normal, shape index and surface reflectance.

<click>
We then compute what we call the effective irradiance, which I will explain in just a moment.

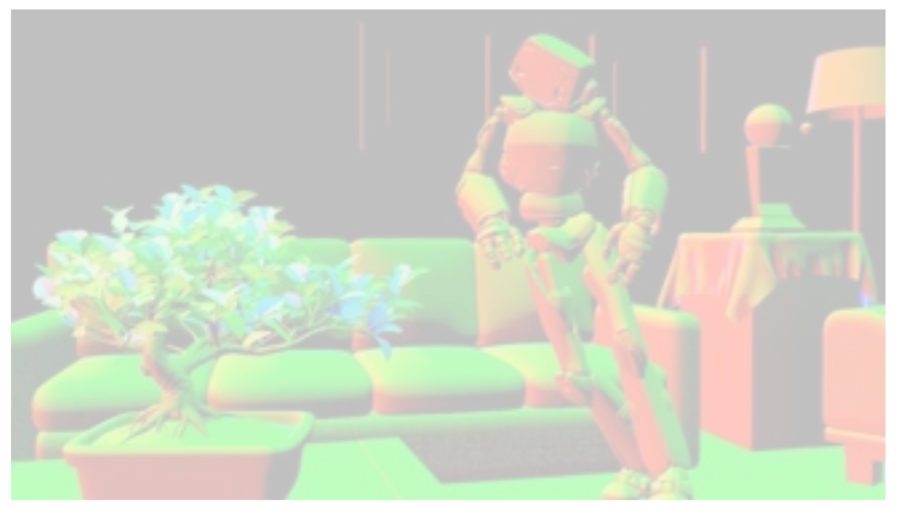
<click>
Finally, we have two motion vector buffers, one that encodes the motion of the geometry in the scene, and one that encodes the motion of the irradiance alone.

Per-component Features

Color

Diffuse Direct

Normal



Shape Index

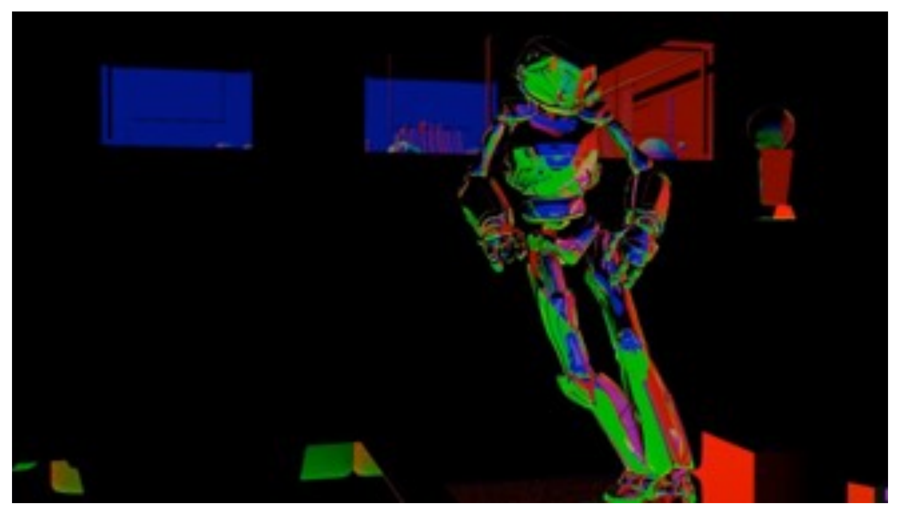
Reflectance

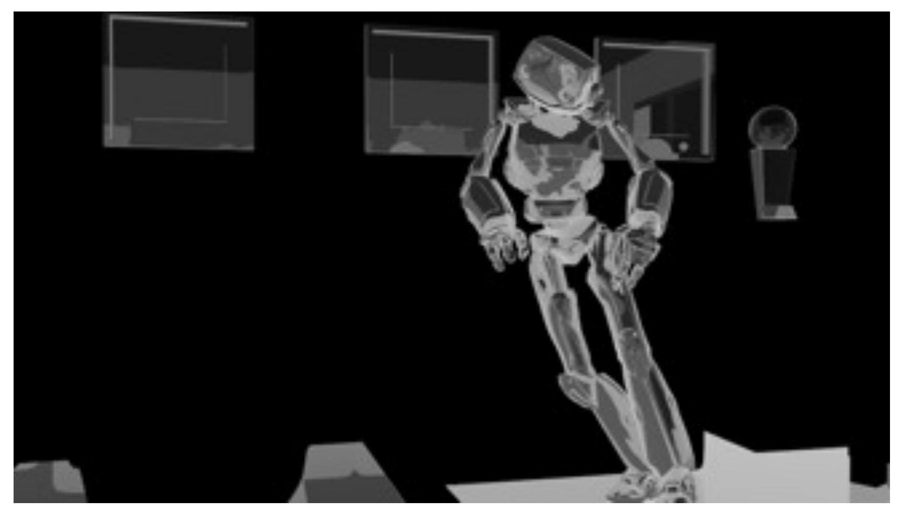
Eff. Irradiance

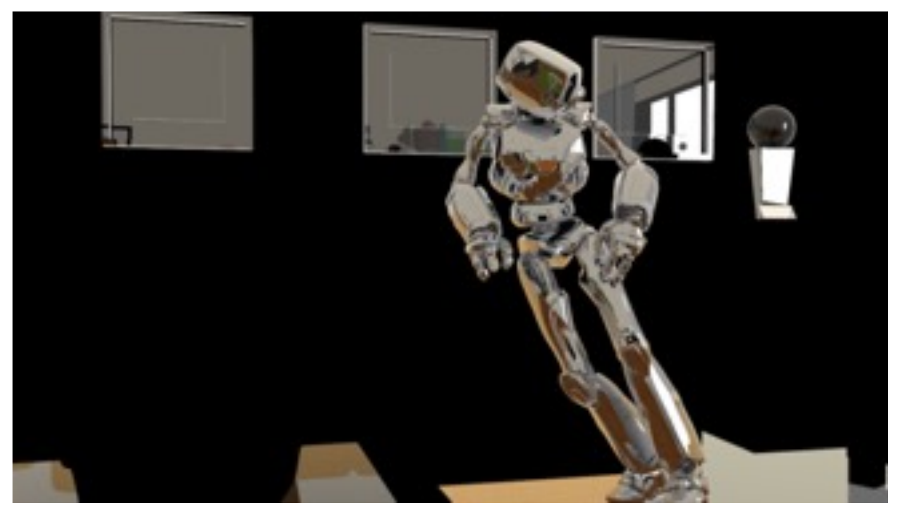
Geom. Motion

Irradiance Motion

Specular Reflection







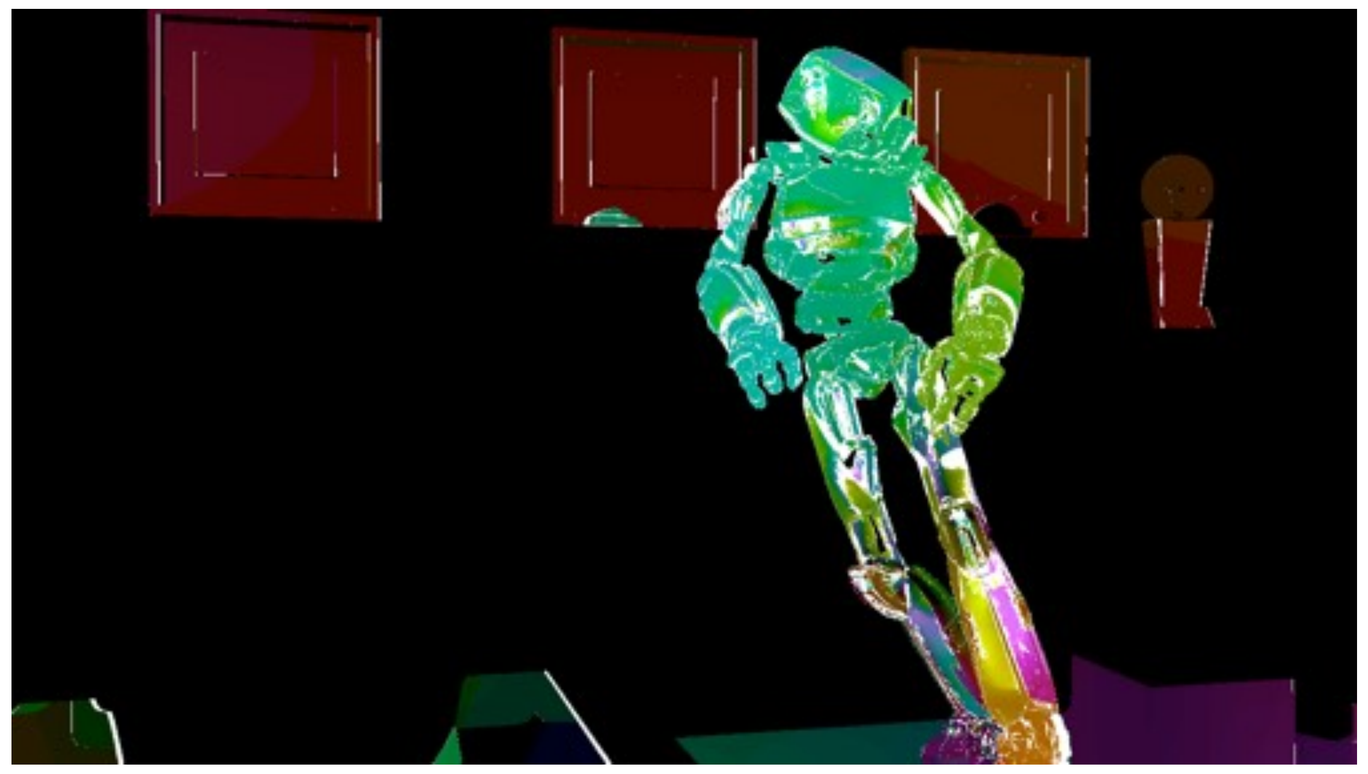
Spec. Motion

This is done for every component of the decomposition.
Here I show the specular reflections, and it works the same way for the other ones.

Novel Features

Effective Irradiance

Specular Motion



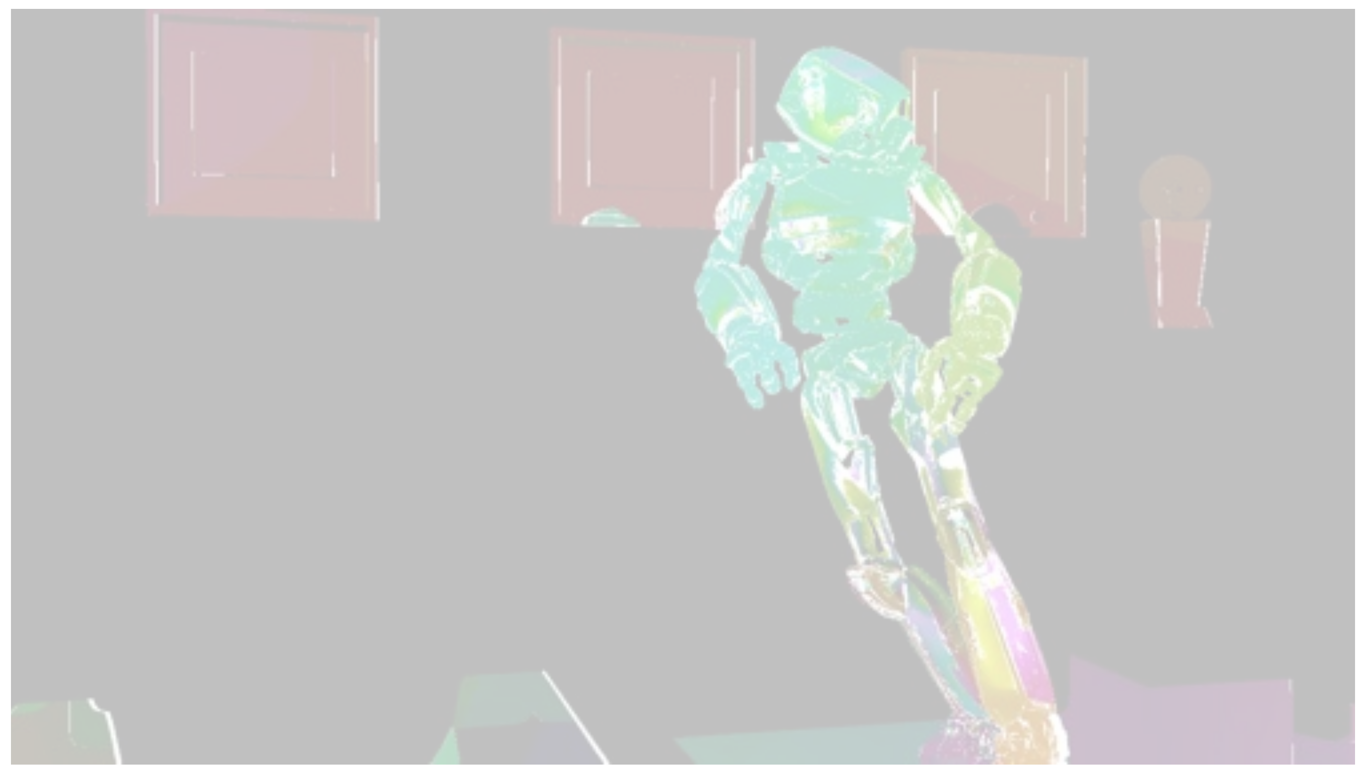
Irradiance Motion

The contributions in our work concern the computation of the three features shown here, which we will now describe in more detail…

Novel Features

Effective Irradiance

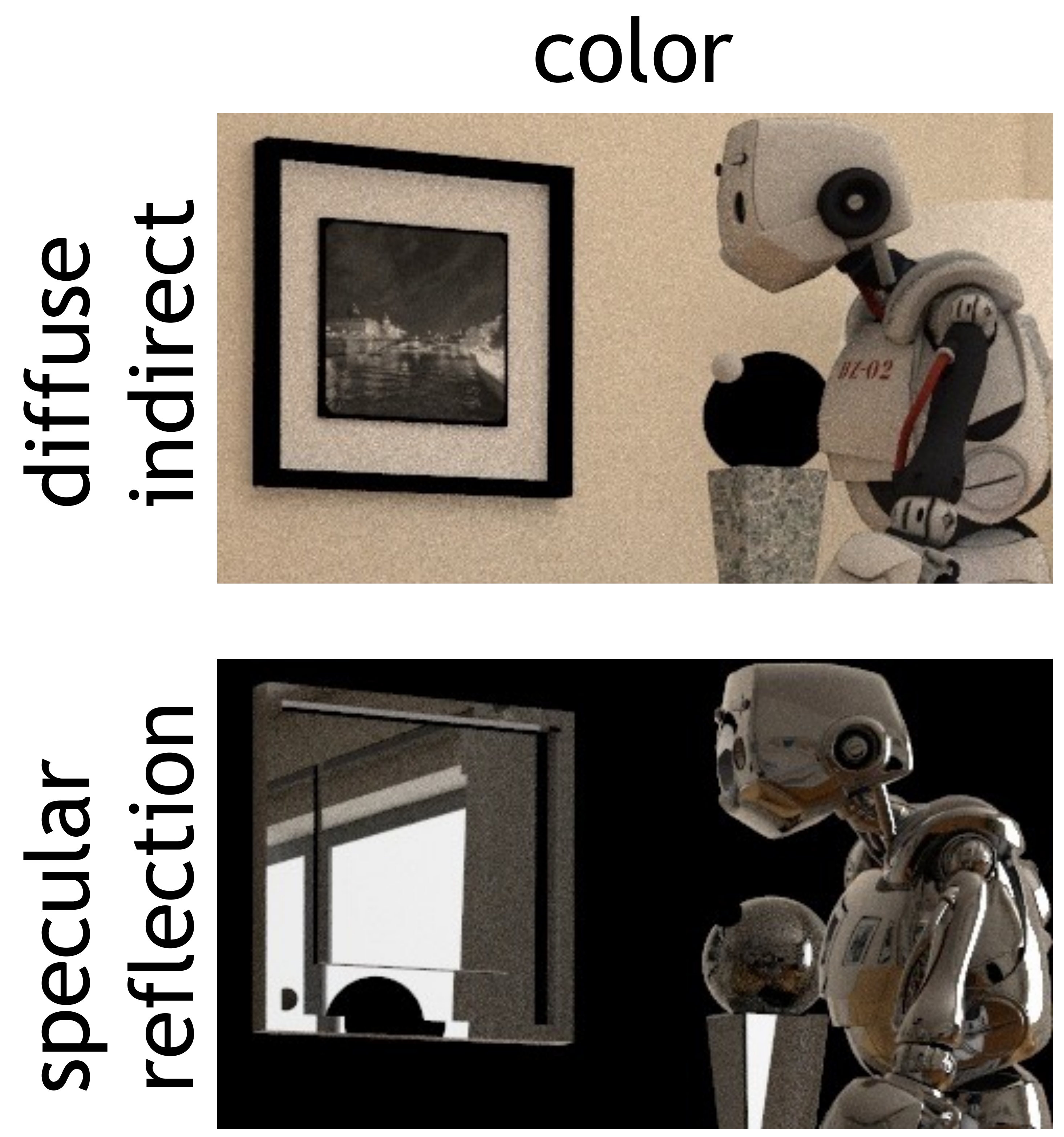
Specular Motion



Irradiance Motion

…starting with our effective irradiance factorization.

Effective Irradiance Factorization



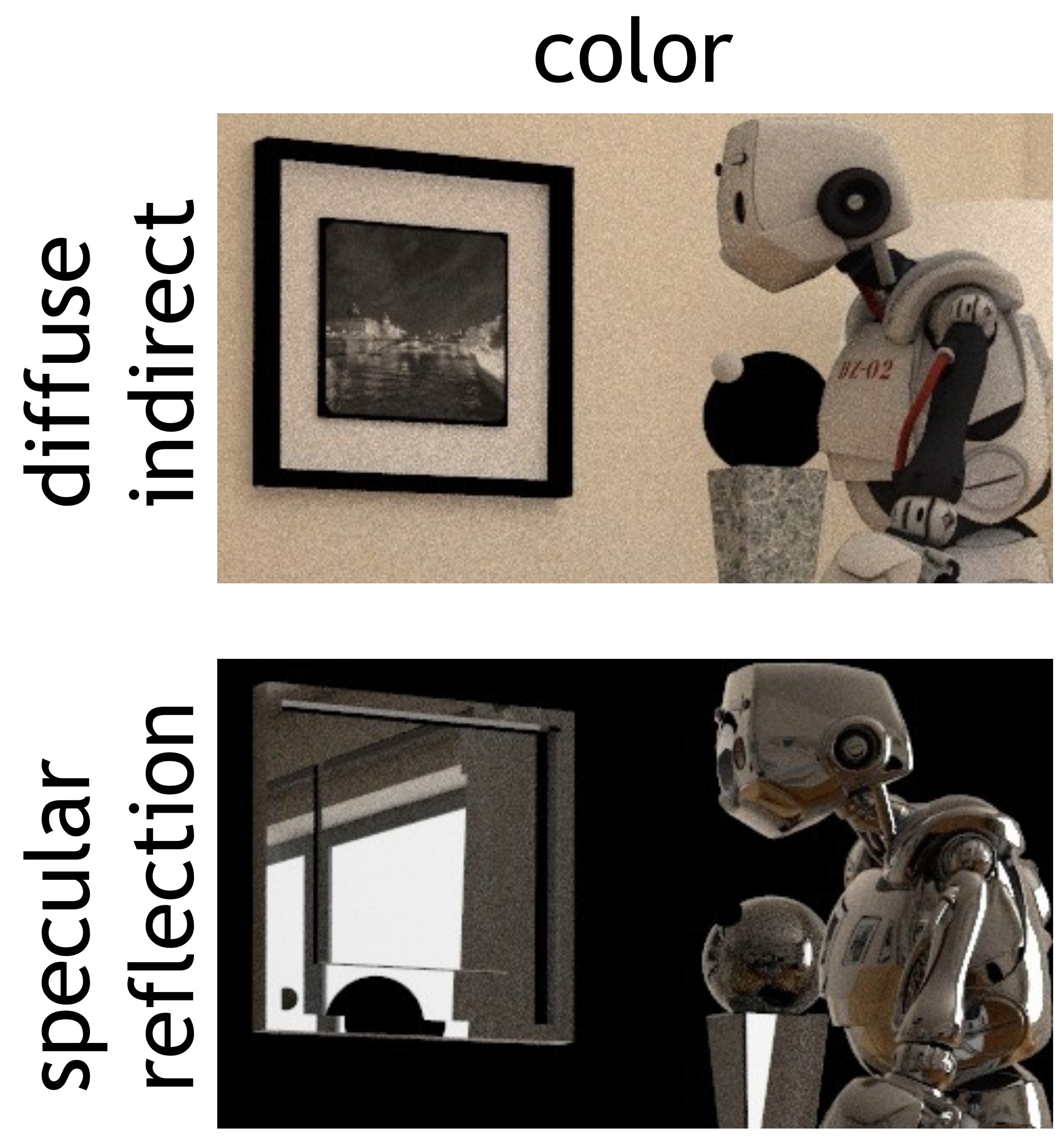
Standard irradiance factorization

- Only Lambertian surfaces
- No motion blur, depth-of-field

The standard irradiance factorization used in previous work is only valid for Lambertian surfaces with no motion blur or depth-of-field.

<click>
However, in this scene, the robot is made of a rough plastic material and the sphere of solid glass, both of which invalidate the standard irradiance factorization assumptions.

Effective Irradiance Factorization



Standard irradiance factorization

Only Lambertian surfaces

No motion blur, depth-of-field

Robot: rough plastic material

diffuse + rough dielectric coating

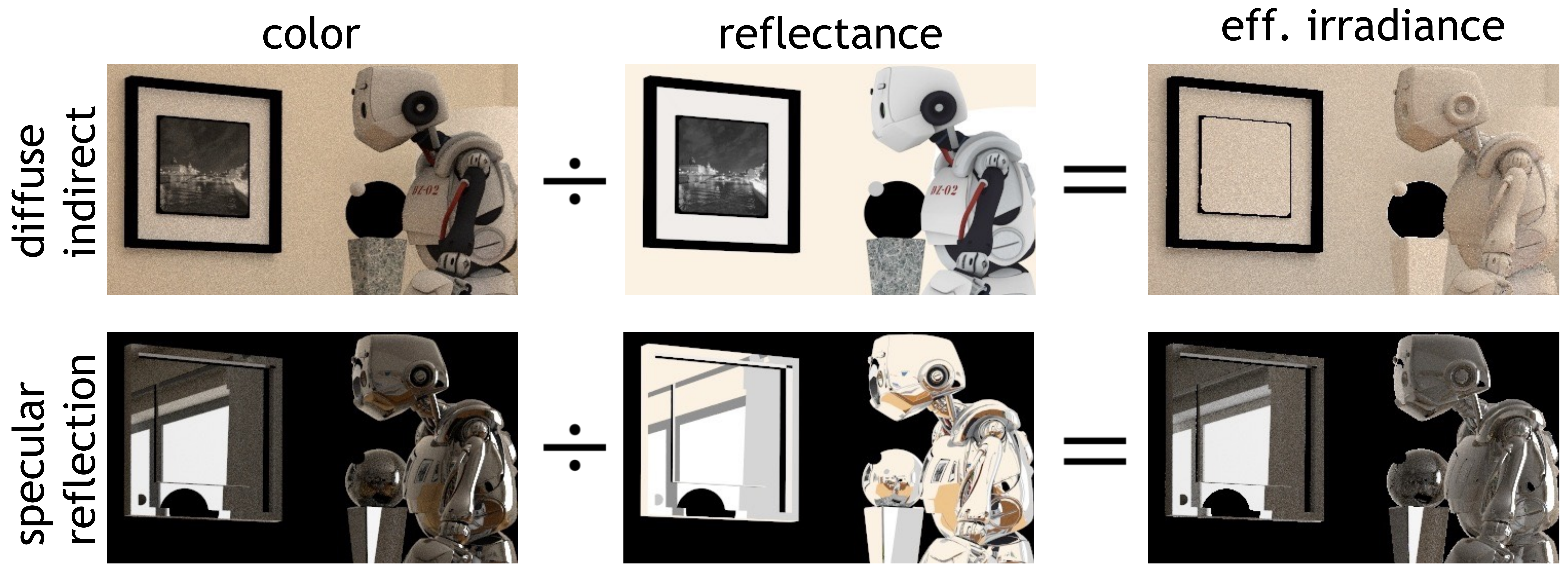
Sphere: solid glass

smooth dielectric

The standard irradiance factorization used in previous work is only valid for Lambertian surfaces with no motion blur or depth-of-field.

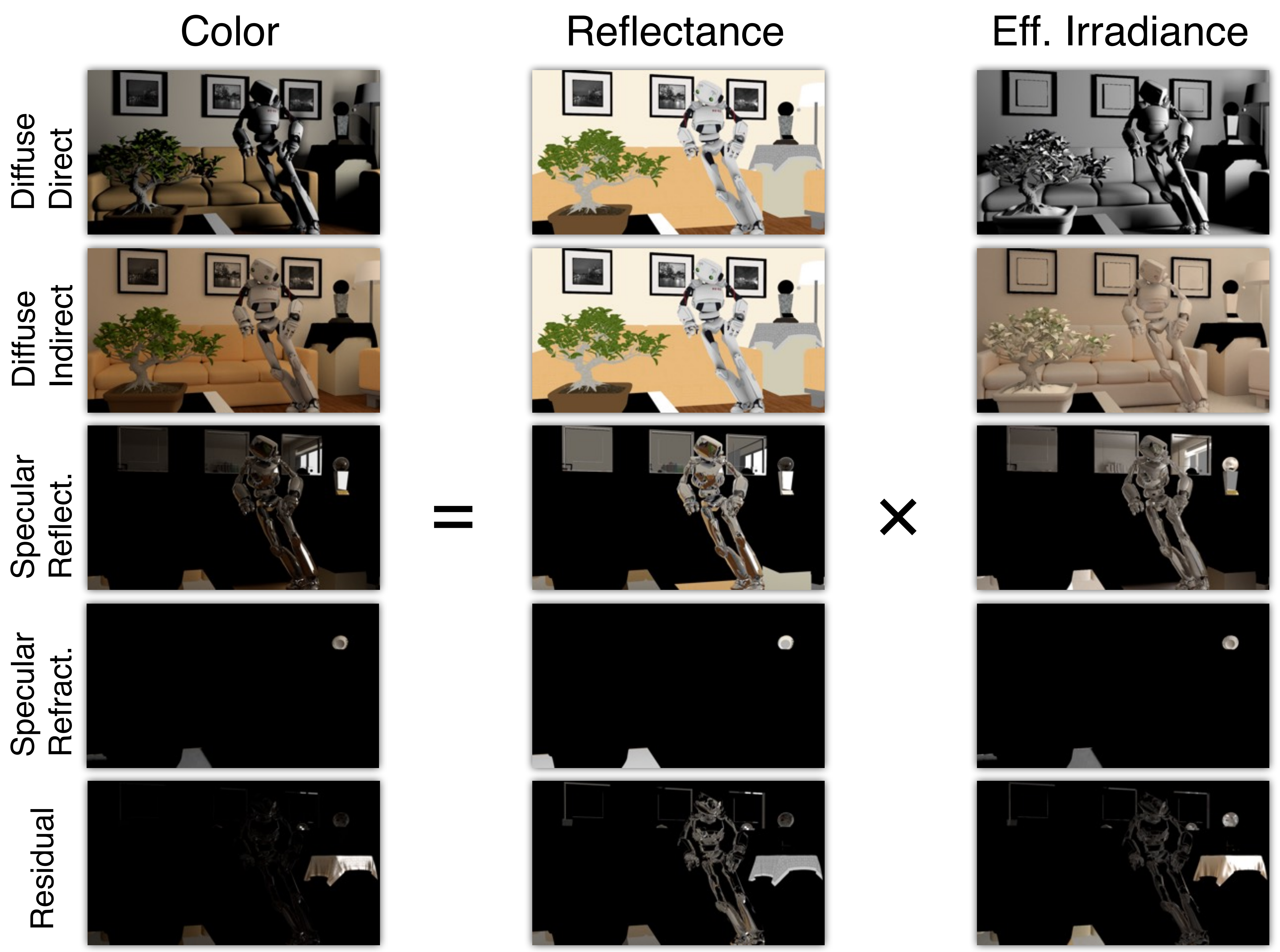
<click>
However, in this scene, the robot is made of a rough plastic material and the sphere of solid glass, both of which invalidate the standard irradiance factorization assumptions.

Effective Irradiance Factorization



In our system, every material, whether Lambertian or not, provides a reflectance value at a given surface point that we store in a buffer.
We then divide the final component pixel color by this reflectance.

We call the result of this the effective irradiance, which is actually identical to the irradiance whenever the surface is Lambertian.

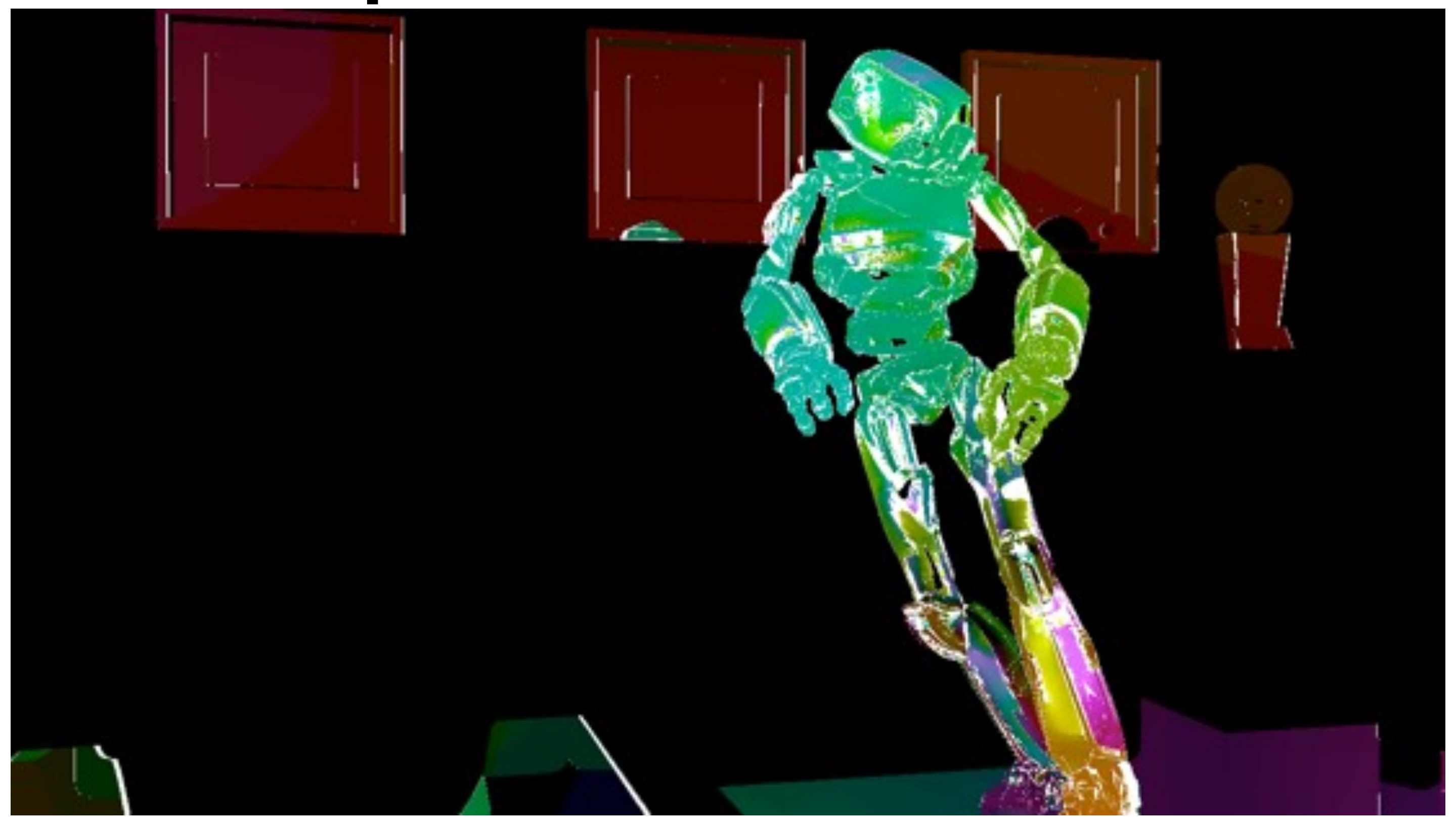


We then apply our effective irradiance factorization to all components of our decomposition, allowing us to separately process the reflectance and irradiance and then multiply the results to get the final pixel color.

Novel Features

Effective Irradiance

Specular Motion

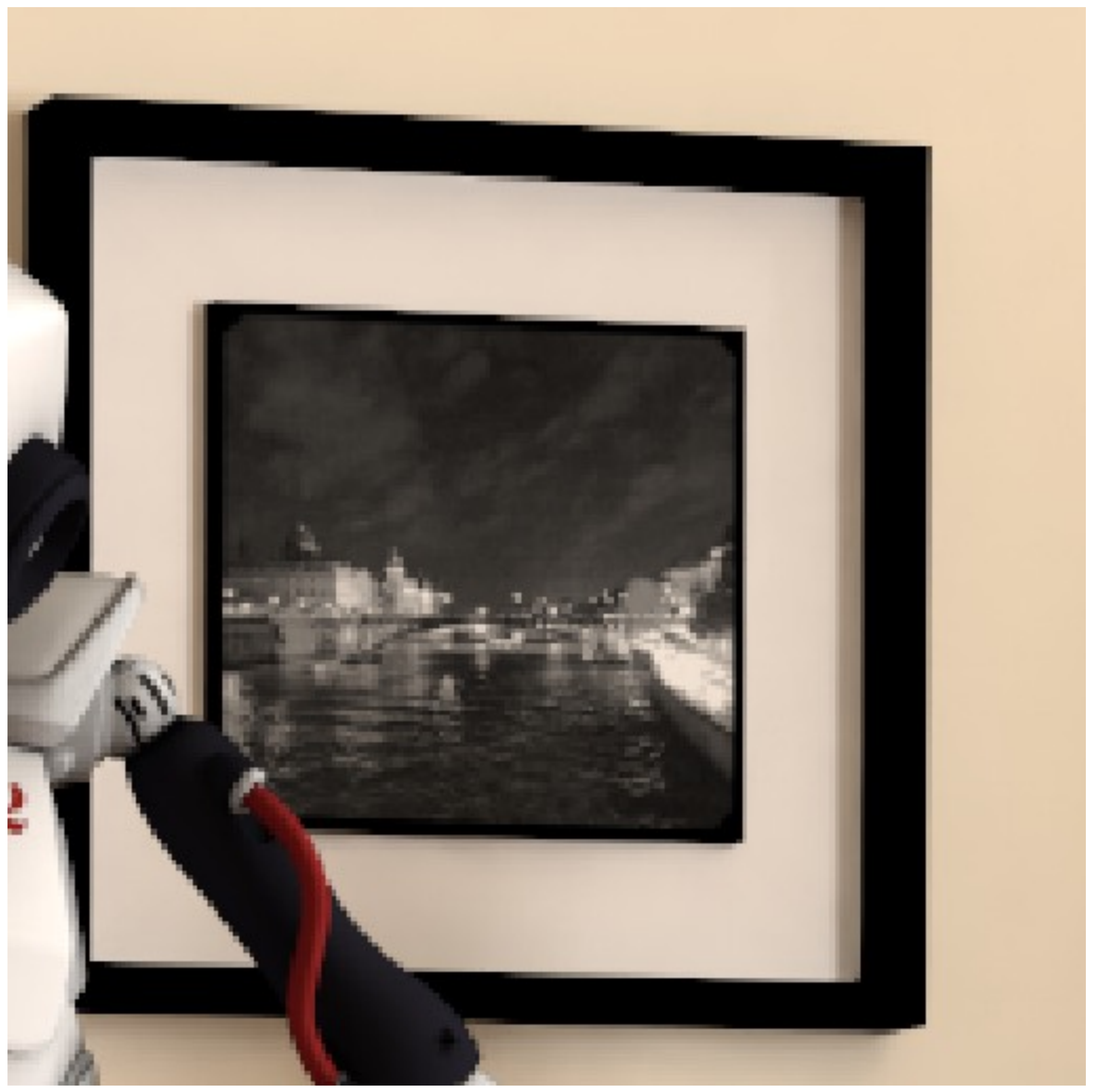


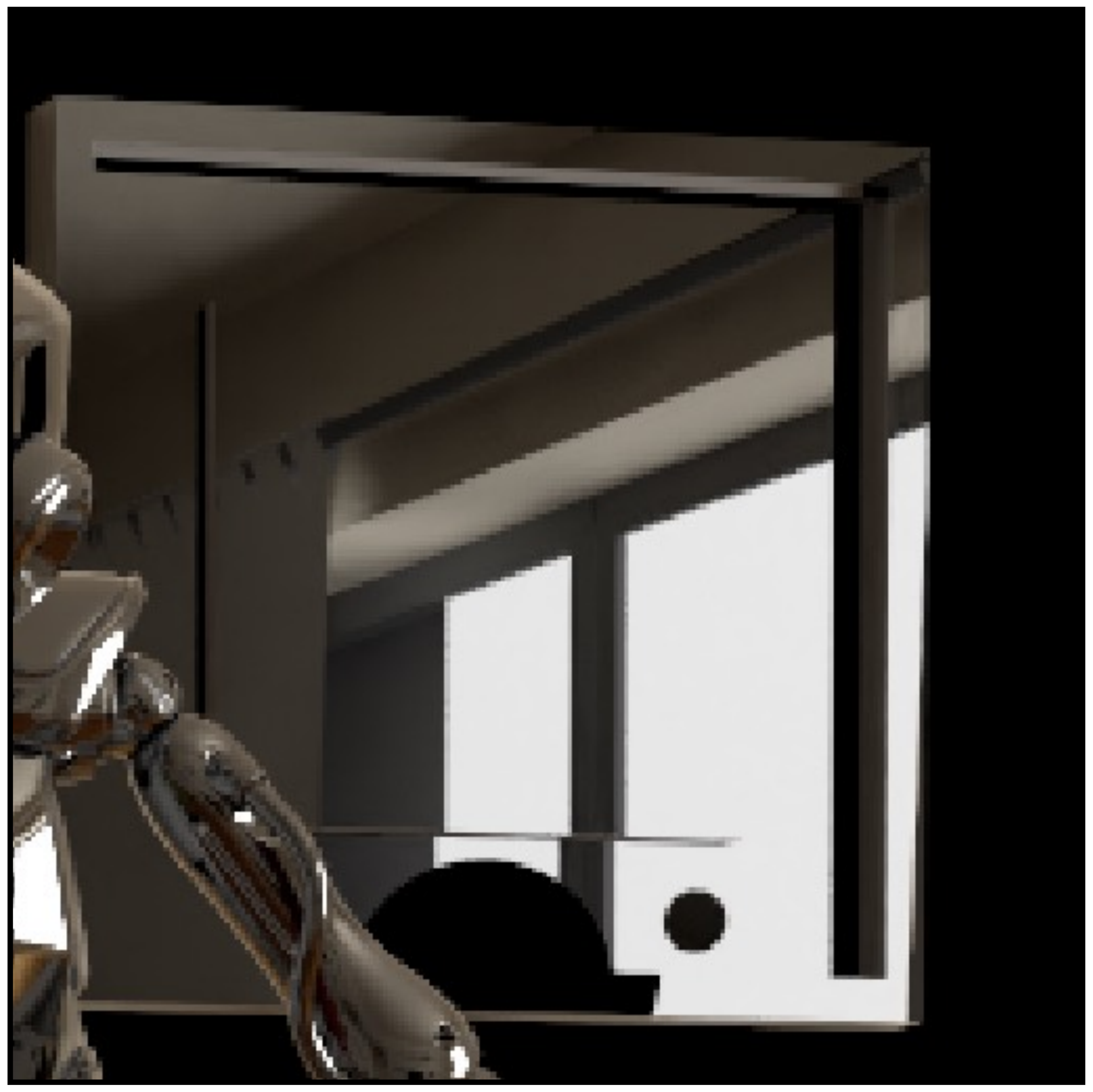
Irradiance Motion

We now turn to the specular motion estimation

WENZEL

Motion Estimation


Diffuse Component


Specular reflection component

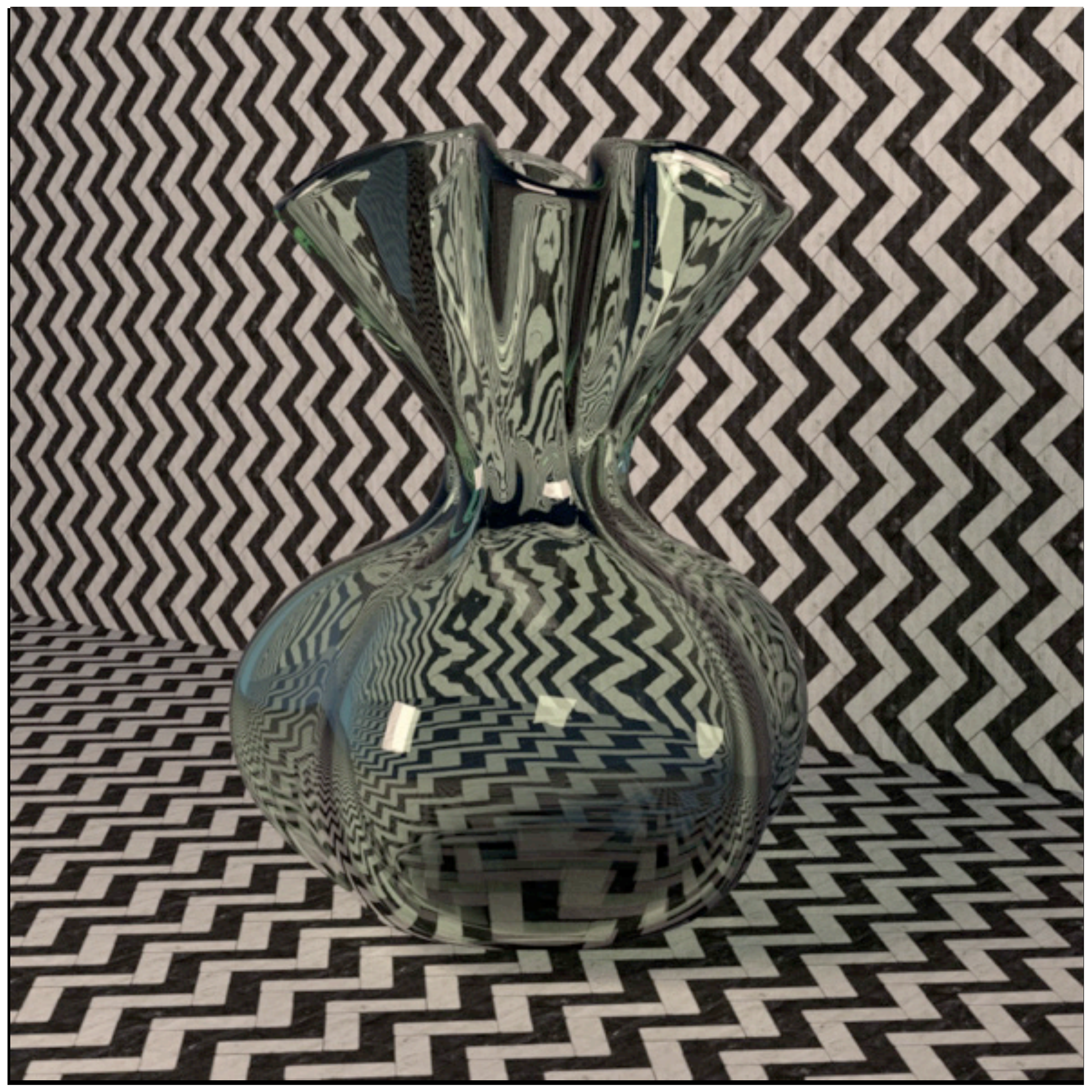
Diffuse component: use geometry motion vectors

Specular components: ???

We compute a complete set of motion vectors for every component in our decomposition. This is simple for the diffuse component, because the renderer exactly knows where a point on an object will be in the next frame. But when specular interactions are involved, things get more tricky -- the observed motion in screen space generally doesn't match the motion of the reflecting object. When we look at a scene through a moving curved glass object, the rendering gets warped in intricate non-linear ways, and in this part of the talk I'll introduce the tools to deal with that.

Motion Estimation

Diffuse Component

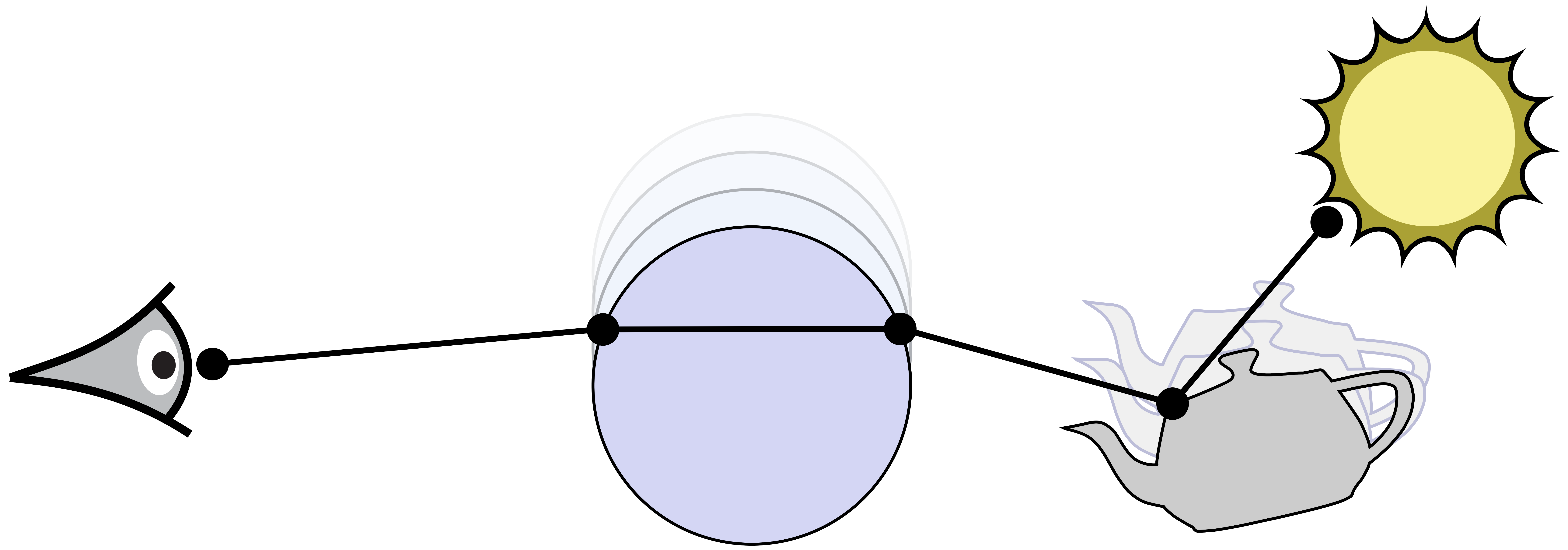

Specular reflection component

Diffuse component: use geometry motion vectors

Specular components: ???

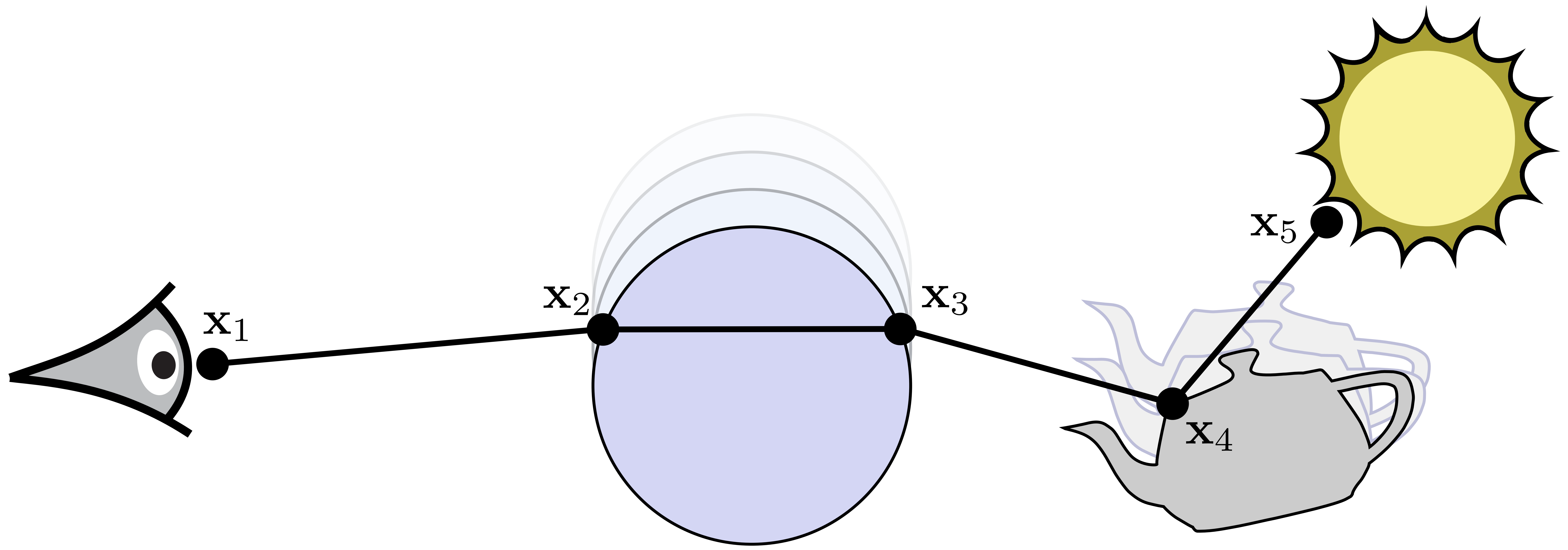
We compute a complete set of motion vectors for every component in our decomposition. This is simple for the diffuse component, because the renderer exactly knows where a point on an object will be in the next frame. But when specular interactions are involved, things get more tricky -- the observed motion in screen space generally doesn’t match the motion of the reflecting object. When we look at a scene through a moving curved glass object, the rendering gets warped in intricate non-linear ways, and in this part of the talk I’ll introduce the tools to deal with that.

Motion vectors for specular paths



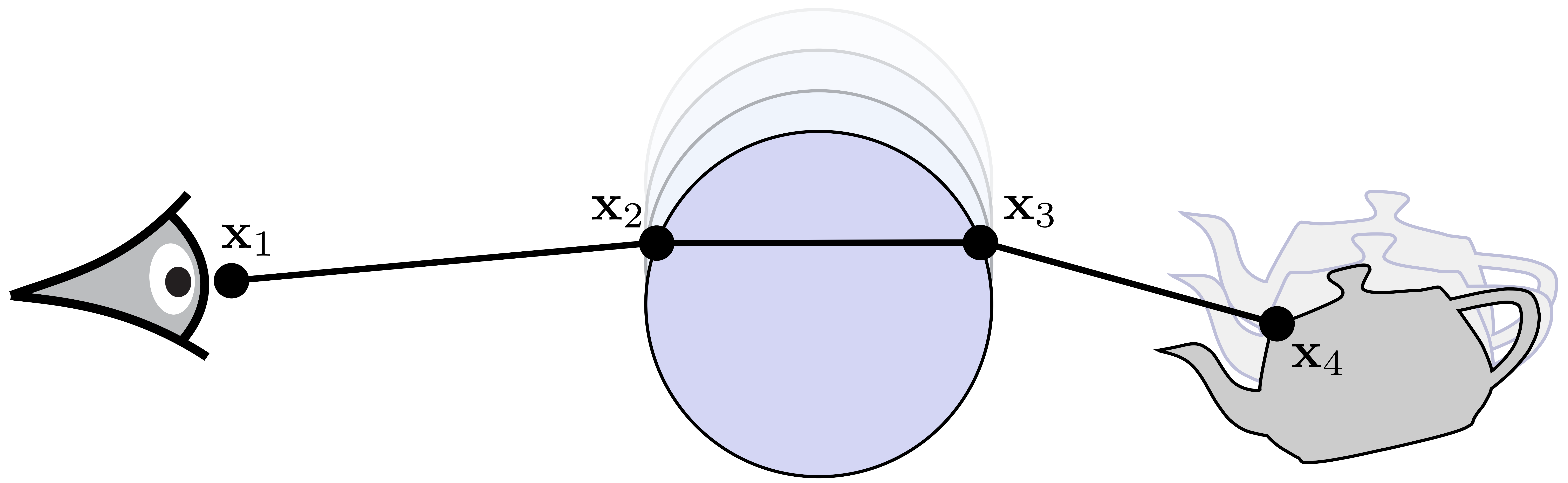
.. so how does that work? Let’s consider this example light path, where an object is seen through a glass sphere, with vertices x1 to x5. We are only interested in the motion of the first non-specular object seen from the camera, so we can ignore later vertices. Our formulation forces the endpoints x_1 and x_4 to stick to the same point on the camera and the teapot, while allowing the intermediate vertices to move freely. We then want to evolve the light path over time to find its configuration at the next frame. One thing to keep in mind is that this problem doesn’t always have a solution -- there can be reflections or refractions that disappear and simply don’t exist anymore in another frame, so we need to be prepared to deal with that. It turns out that this is quite related to the manifold exploration technique, and we can solve it by adapting its tools.

Motion vectors for specular paths



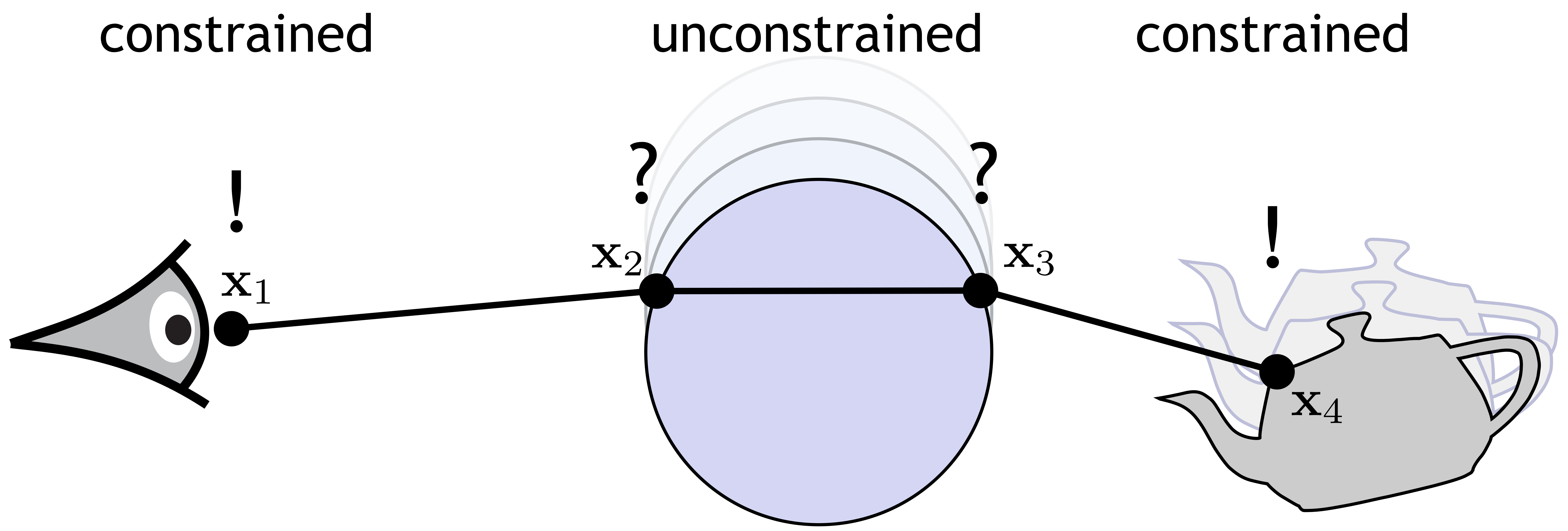
.. so how does that work? Let’s consider this example light path, where an object is seen through a glass sphere, with vertices x1 to x5. We are only interested in the motion of the first non-specular object seen from the camera, so we can ignore later vertices. Our formulation forces the endpoints x_1 and x_4 to stick to the same point on the camera and the teapot, while allowing the intermediate vertices to move freely. We then want to evolve the light path over time to find its configuration at the next frame. One thing to keep in mind is that this problem doesn’t always have a solution -- there can be reflections or refractions that disappear and simply don’t exist anymore in another frame, so we need to be prepared to deal with that.   It turns out that this is quite related to the manifold exploration technique, and we can solve it by adapting its tools.

Motion vectors for specular paths



.. so how does that work? Let’s consider this example light path, where an object is seen through a glass sphere, with vertices x1 to x5. We are only interested in the motion of the first non-specular object seen from the camera, so we can ignore later vertices. Our formulation forces the endpoints x_1 and x_4 to stick to the same point on the camera and the teapot, while allowing the intermediate vertices to move freely. We then want to evolve the light path over time to find its configuration at the next frame. One thing to keep in mind is that this problem doesn’t always have a solution -- there can be reflections or refractions that disappear and simply don’t exist anymore in another frame, so we need to be prepared to deal with that. It turns out that this is quite related to the manifold exploration technique, and we can solve it by adapting its tools.

Motion vectors for specular paths

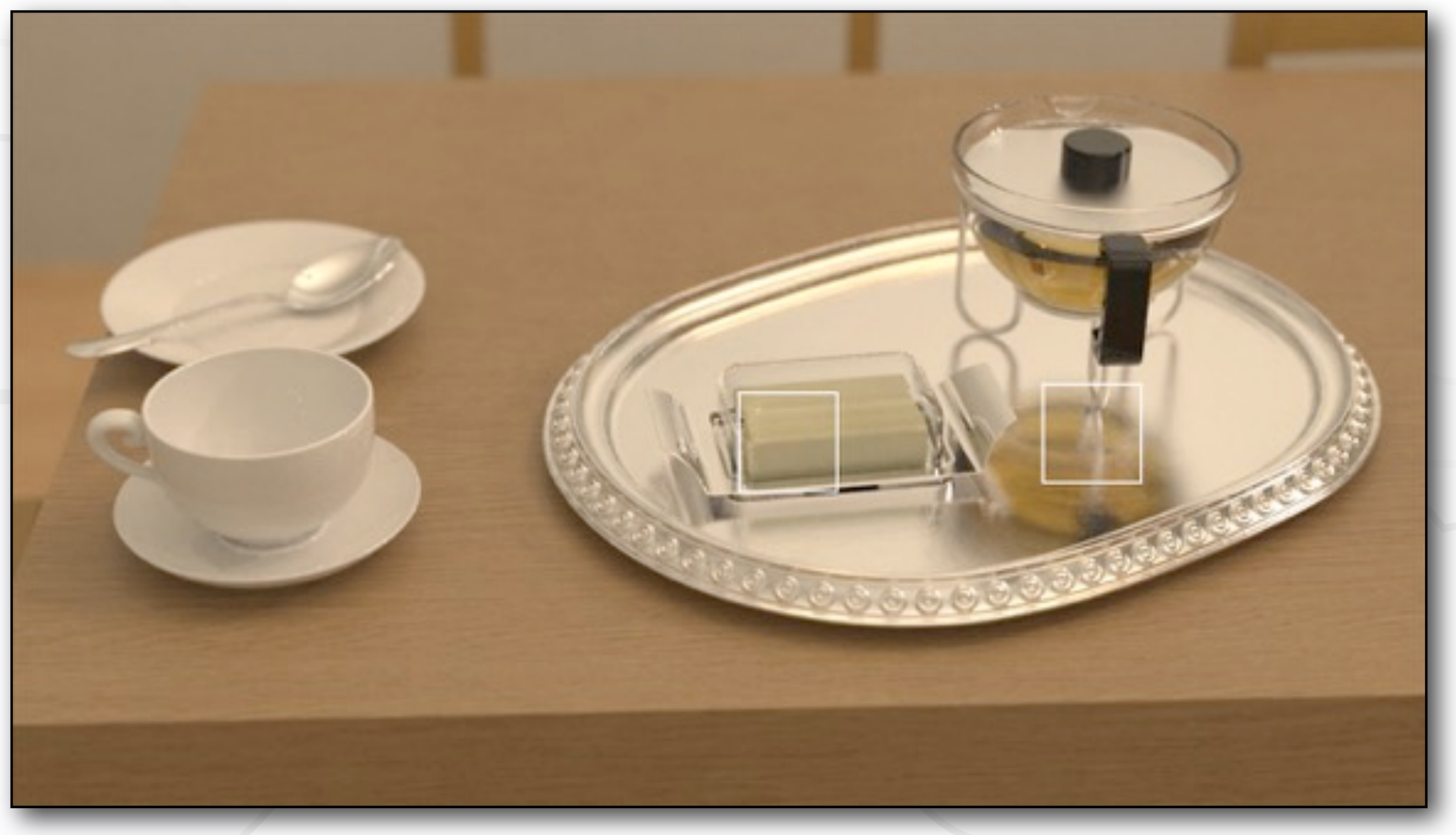


.. so how does that work? Let’s consider this example light path, where an object is seen through a glass sphere, with vertices x1 to x5. We are only interested in the motion of the first non-specular object seen from the camera, so we can ignore later vertices. Our formulation forces the endpoints x_1 and x_4 to stick to the same point on the camera and the teapot, while allowing the intermediate vertices to move freely. We then want to evolve the light path over time to find its configuration at the next frame. One thing to keep in mind is that this problem doesn’t always have a solution -- there can be reflections or refractions that disappear and simply don’t exist anymore in another frame, so we need to be prepared to deal with that.   It turns out that this is quite related to the manifold exploration technique, and we can solve it by adapting its tools.

Motion vectors for specular paths

This problem can be solved using specular manifolds!

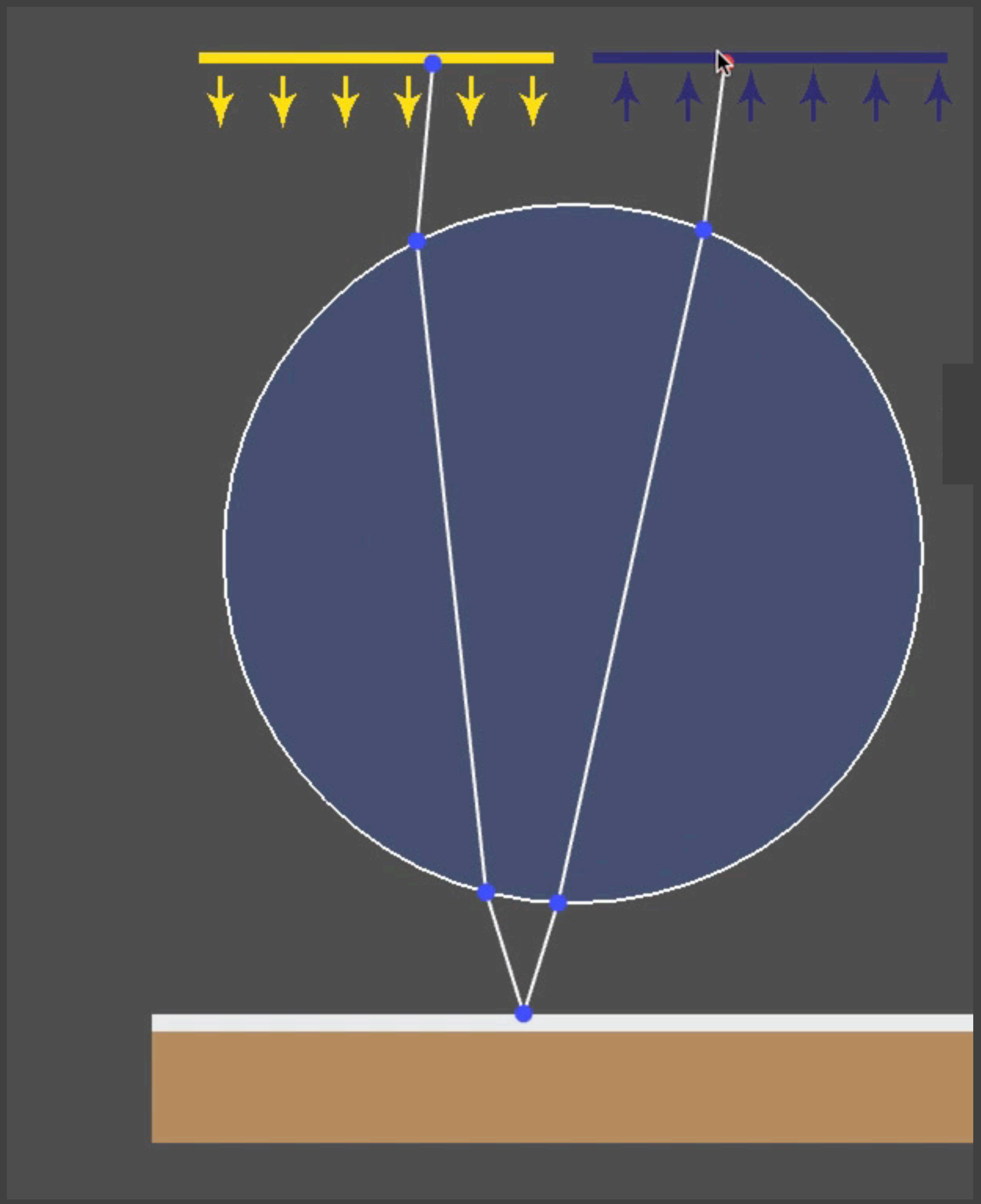
constrained unconstrained constrained

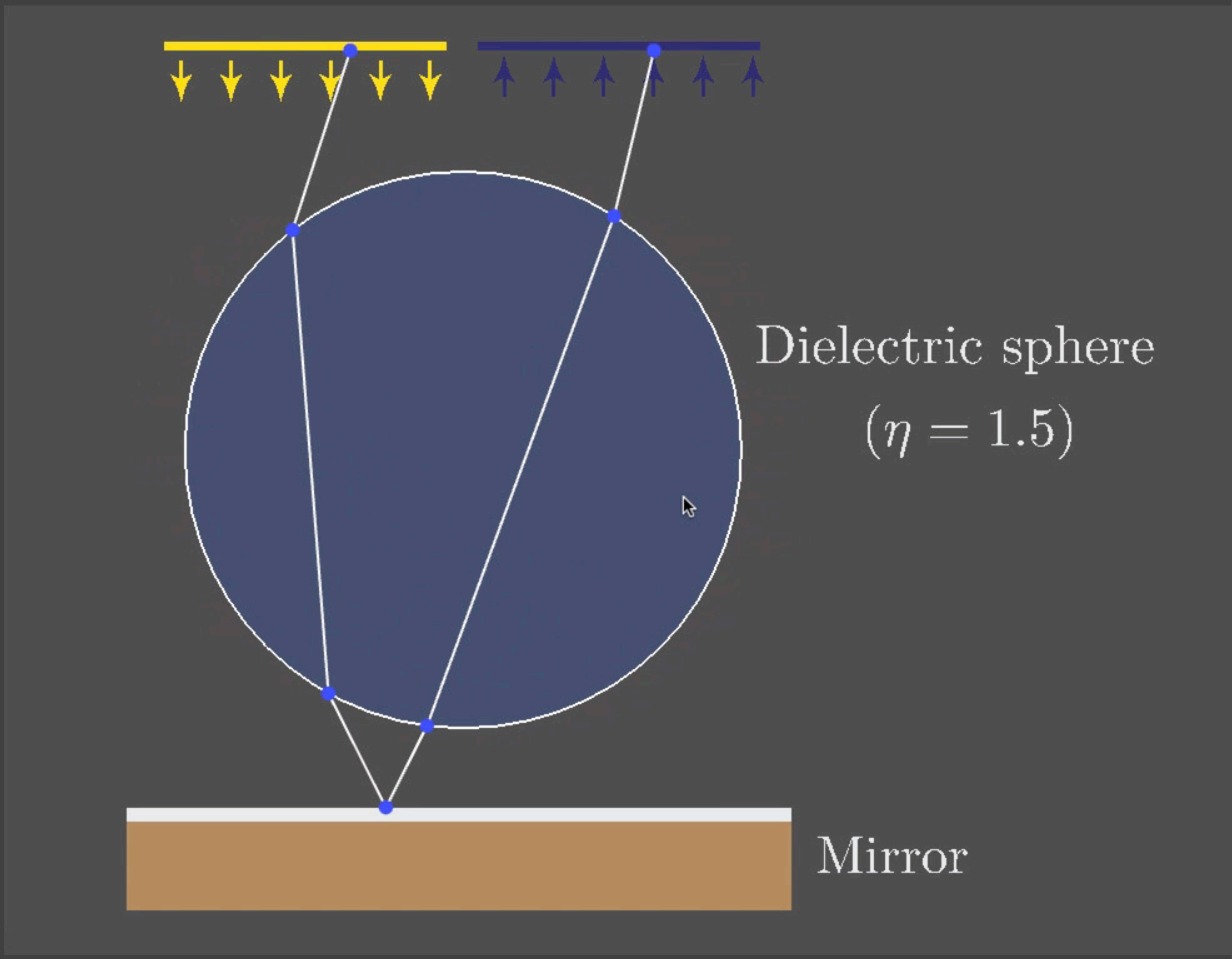


[Jakob and Marschner 12]

Monday, June 29, 15

.. so how does that work? Let’s consider this example light path, where an object is seen through a glass sphere, with vertices x1 to x5. We are only interested in the motion of the first non-specular object seen from the camera, so we can ignore later vertices. Our formulation forces the endpoints x_1 and x_4 to stick to the same point on the camera and the teapot, while allowing the intermediate vertices to move freely. We then want to evolve the light path over time to find its configuration at the next frame. One thing to keep in mind is that this problem doesn’t always have a solution -- there can be reflections or refractions that disappear and simply don’t exist anymore in another frame, so we need to be prepared to deal with that. It turns out that this is quite related to the manifold exploration technique, and we can solve it by adapting its tools.





Manifold walk
[Jakob and Marschner 12]

Temporal manifold walk
This paper

Here is a graphical illustration of the differences.

In the original manifold exploration paper, the scene was assumed to be static, and we were solving for a valid light path through a chain of specular reflections or refractions. You can see that on the left, where the red vertex is being moved around, while the manifold walk continually searches for an updated configuration.

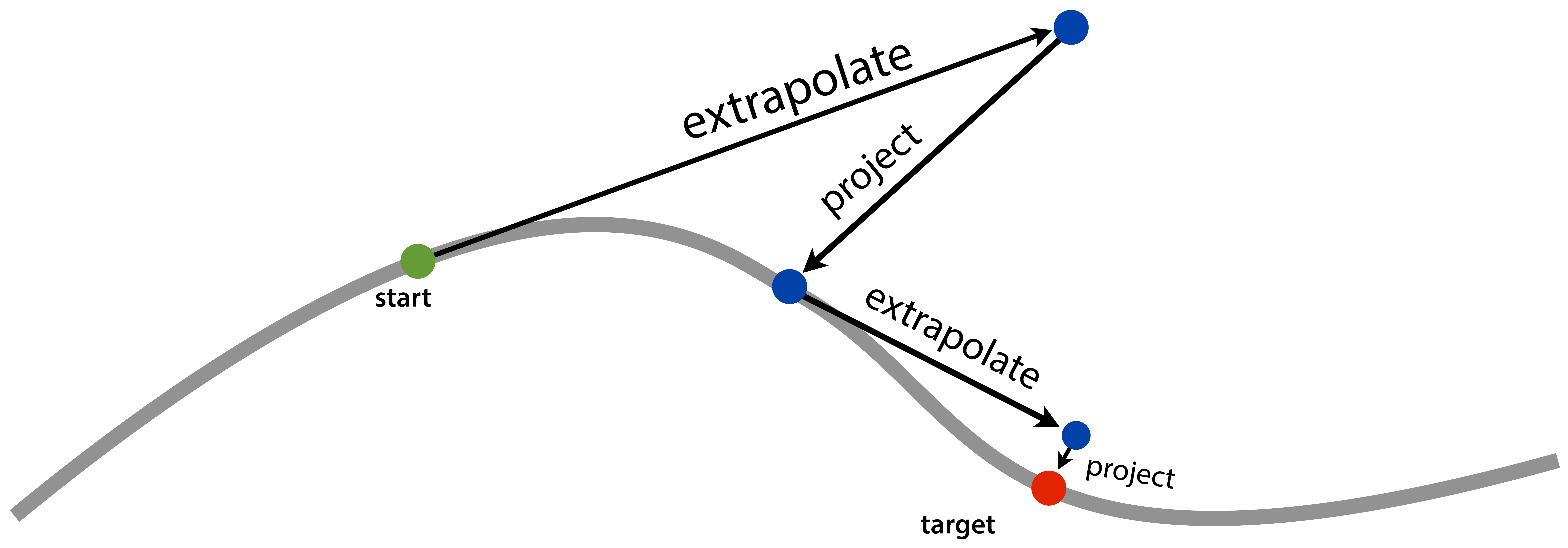
Now, the scene itself is dynamic, and we want to track a single light path over time. That's shown on the right, where the sphere is moving and we solve for a path with the same endpoints. Of course, everything including the endpoints is allowed to move at the same time, we're not restricted to just one object.

Manifold Walk

Basic idea:

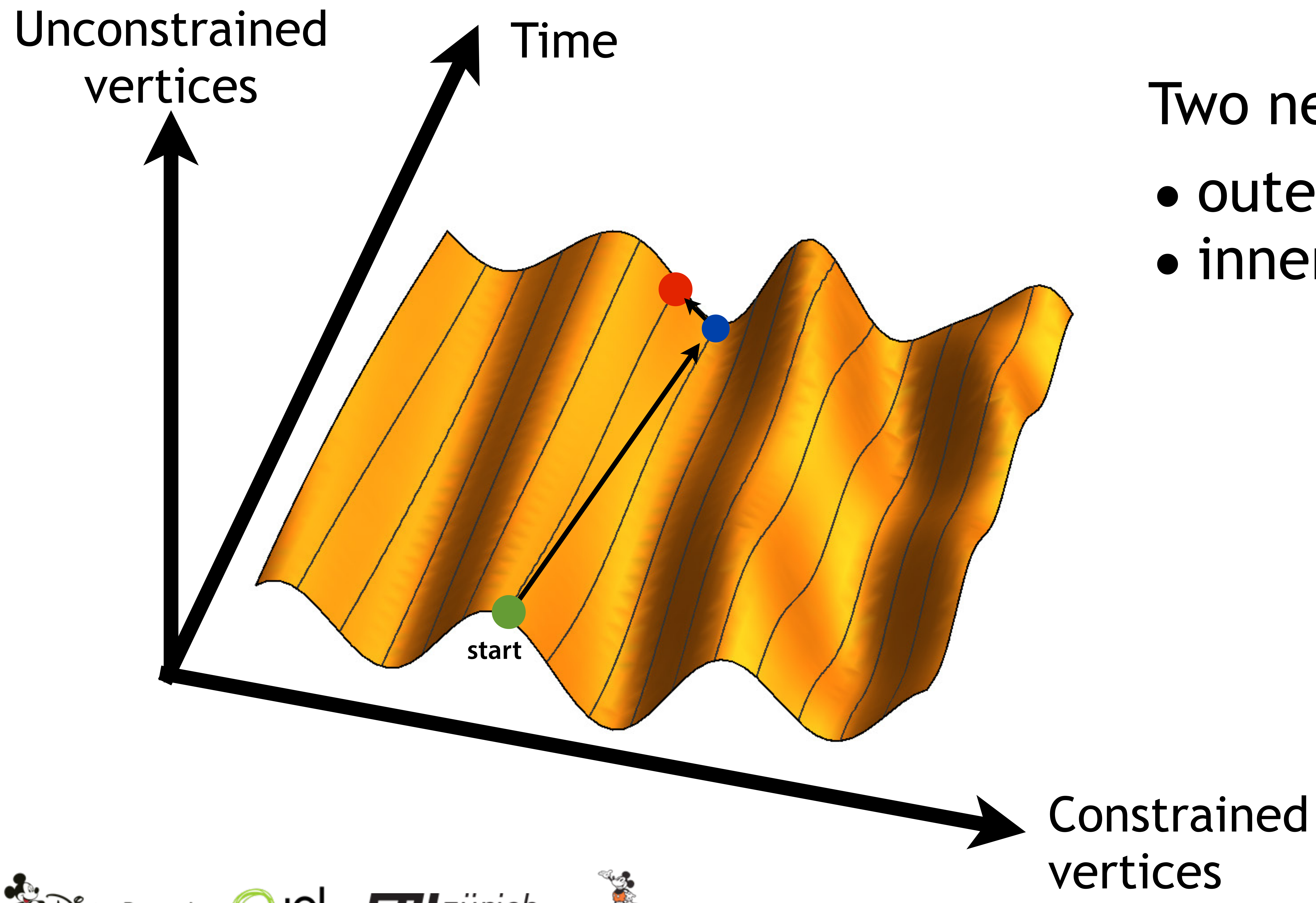
while not there yet:

1. **EXTRAPOLATE**
2. **PROJECT**



The original manifold walk provides a local parameterization of a high-dimensional implicitly defined manifold of valid light paths using a combination of linear extrapolation and projection steps. Repetition of those two steps led to to an algorithm reminiscent of Newton's method with quadratic convergence close to the solution.

Manifold Walks with time



Two nested manifold walks

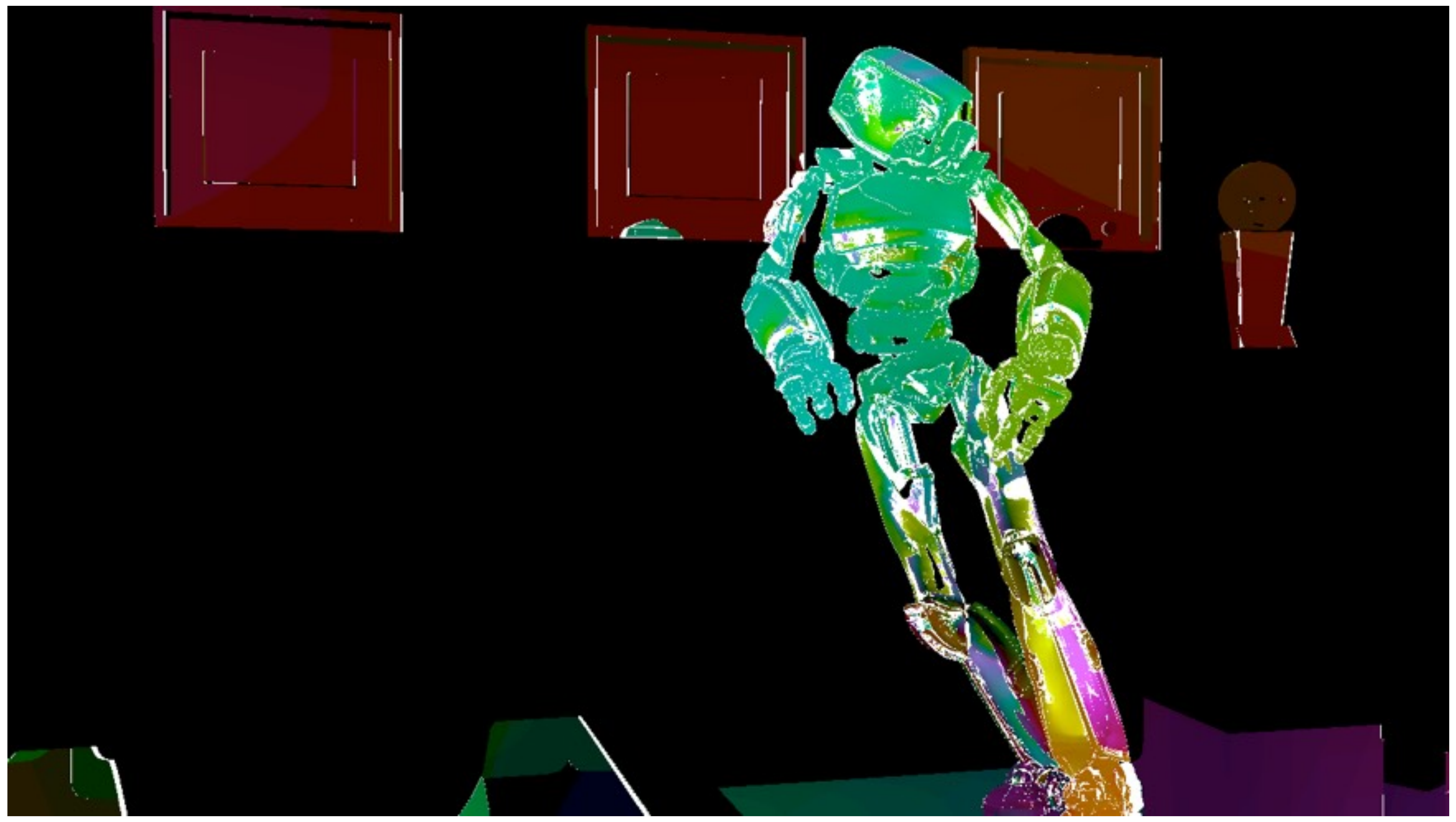
- outer: time steps
- inner: spatial steps

In this paper, we extend specular manifolds with an additional temporal dimension. As mentioned before, we constrain some of the vertices and then the goal is to map the path forward in time -- this corresponds to walking along a iso-parameter line of the manifold, which we do using two nested manifold walks. One takes steps along the time dimension, and another one which maintains the spatial constraints.

Details and Limitations

Needs only local information:

- Positions
- Time derivative of positions
- Normals
- Curvature
- Rel. index of refraction



Altogether, we require remarkably little information to be able to do this. The algorithm just needs local information at the vertices which is generally already available in state of the art rendering systems. When the iteration fails, we tag pixels to force them to be re-rendered later on in a sparse rendering pass.
The implementation is fast, with computation time around a few seconds per frame, and it produces highly accurate motion vectors on the order of the machine precision.
We provide a short self-contained plugin in the Mitsuba repository which computes motion vectors through arbitrary specular chains.

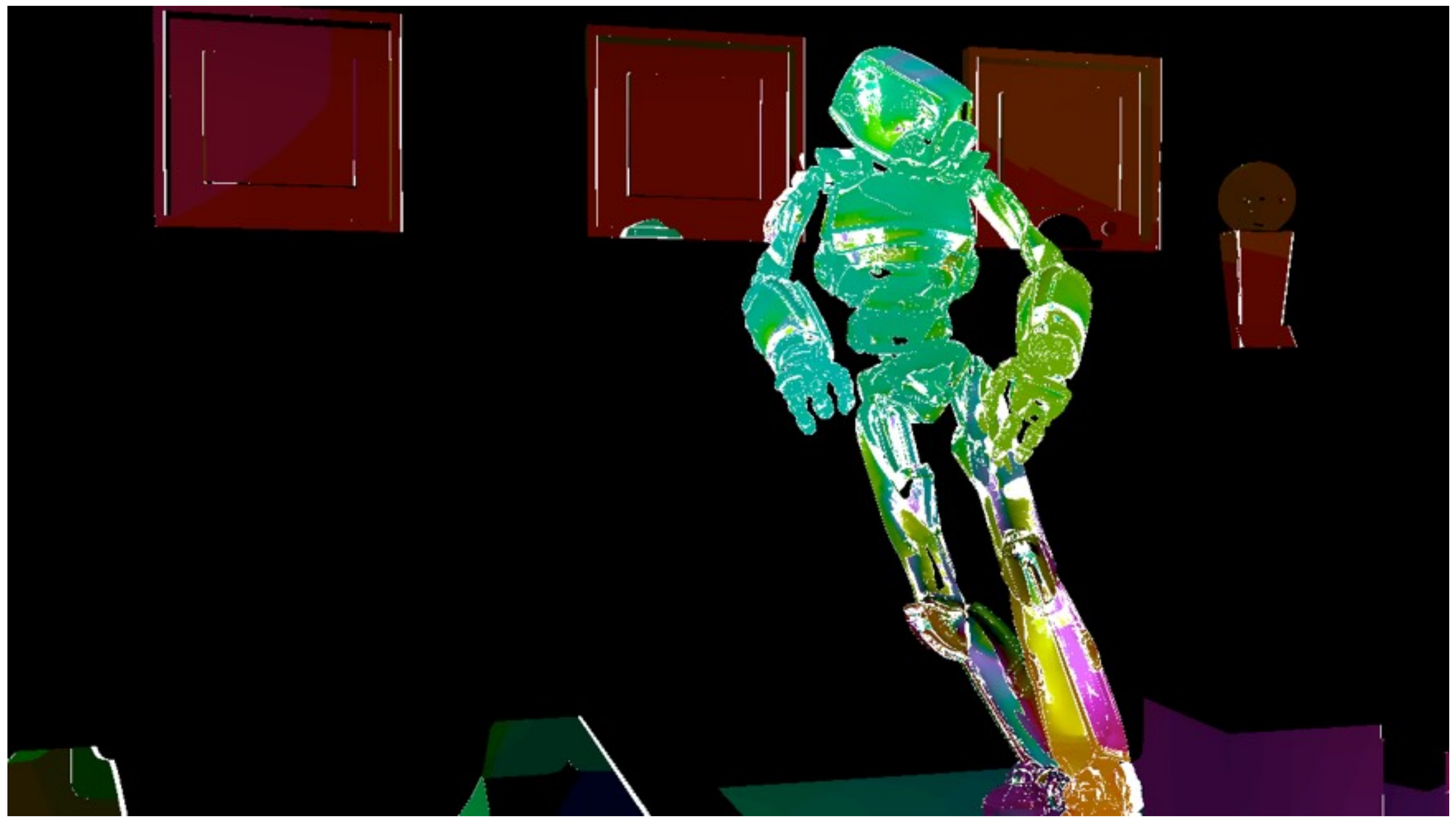
Details and Limitations

Needs only local information:

- Positions
- Time derivative of positions
- Normals
- Curvature
- Rel. index of refraction

Can fail:

- Path ceases to exist
- Path cannot be tracked



Altogether, we require remarkably little information to be able to do this. The algorithm just needs local information at the vertices which is generally already available in state of the art rendering systems. When the iteration fails, we tag pixels to force them to be re-rendered later on in a sparse rendering pass.
The implementation is fast, with computation time around a few seconds per frame, and it produces highly accurate motion vectors on the order of the machine precision.
We provide a short self-contained plugin in the Mitsuba repository which computes motion vectors through arbitrary specular chains.

Details and Limitations

Needs only local information:

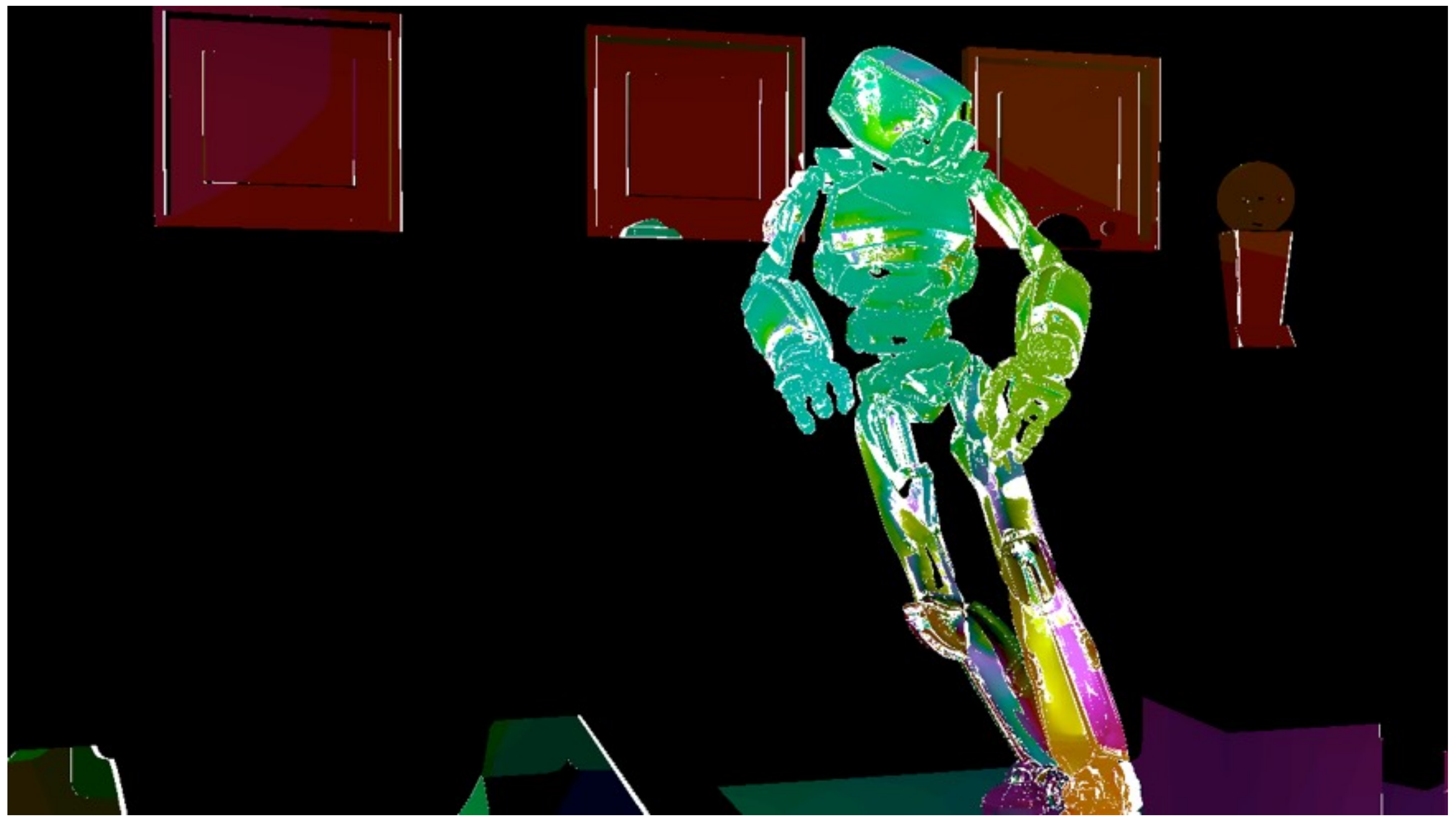
- Positions
- Time derivative of positions
- Normals
- Curvature
- Rel. index of refraction

Fast, simple implementation

- Seconds/frame
- ~500 LOC plugin in Mitsuba (`motion`)

Can fail:

- Path ceases to exist
- Path cannot be tracked

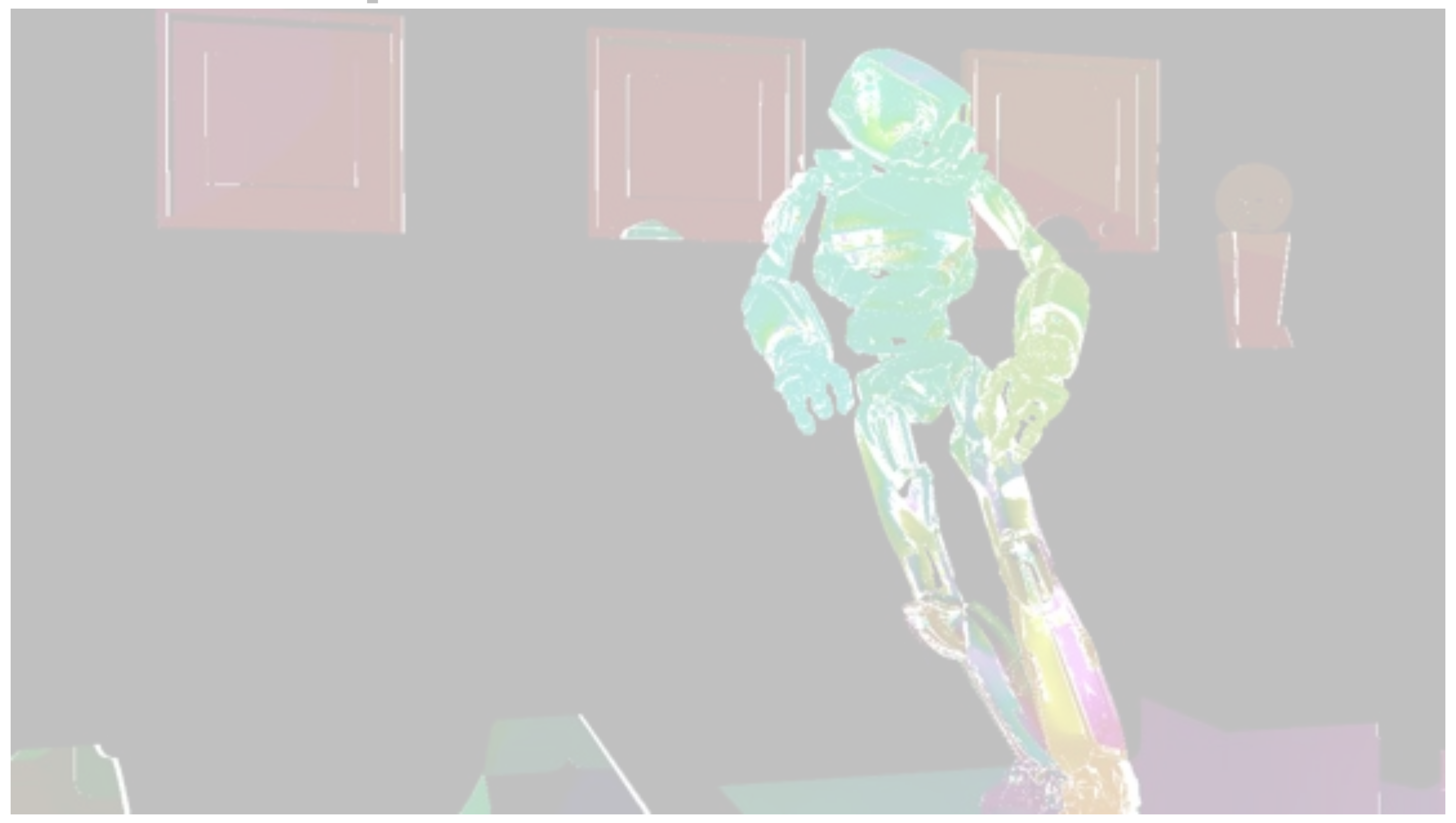


Altogether, we require remarkably little information to be able to do this. The algorithm just needs local information at the vertices which is generally already available in state of the art rendering systems. When the iteration fails, we tag pixels to force them to be re-rendered later on in a sparse rendering pass.
The implementation is fast, with computation time around a few seconds per frame, and it produces highly accurate motion vectors on the order of the machine precision.
We provide a short self-contained plugin in the Mitsuba repository which computes motion vectors through arbitrary specular chains.

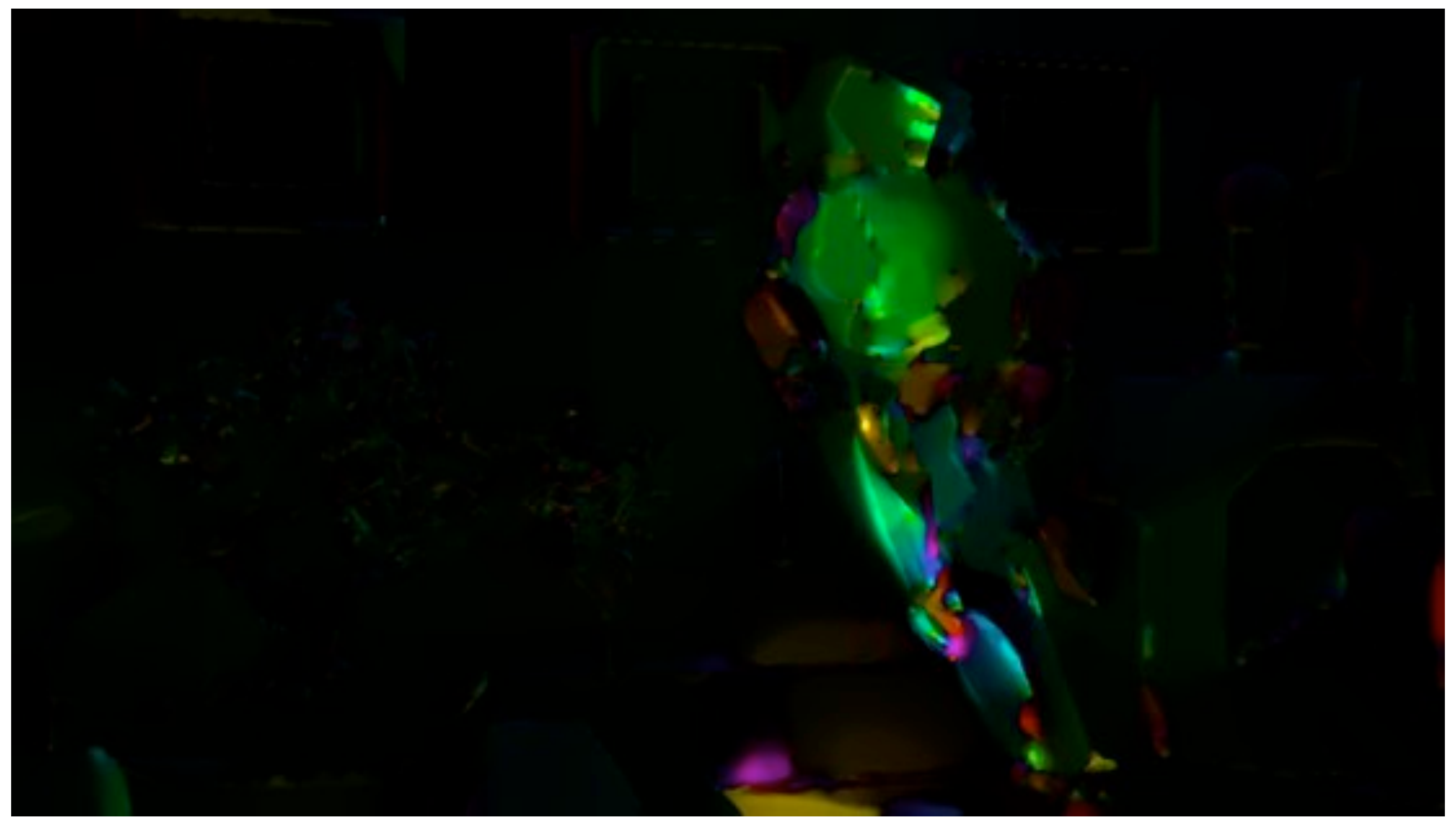
Novel Features

Effective Irradiance

Specular Motion



Irradiance Motion



HENNING

Irradiance Motion Estimation

Frame 1

Frame 2

Direct irradiance component

In addition to the specular components, we also estimate motion vectors for the diffuse irradiance components, which allows us to handle moving shadows.

For the irradiance motion estimation we rely on classic optical flow.

<click>
But we bootstrap the flow computation with the geometry motion vectors to better handle large displacements.

Irradiance Motion Estimation

Frame 1

Frame 2

Direct irradiance component

Bootstrap optical flow with geometry motion vectors

In addition to the specular components, we also estimate motion vectors for the diffuse irradiance components, which allows us to handle moving shadows.

For the irradiance motion estimation we rely on classic optical flow.

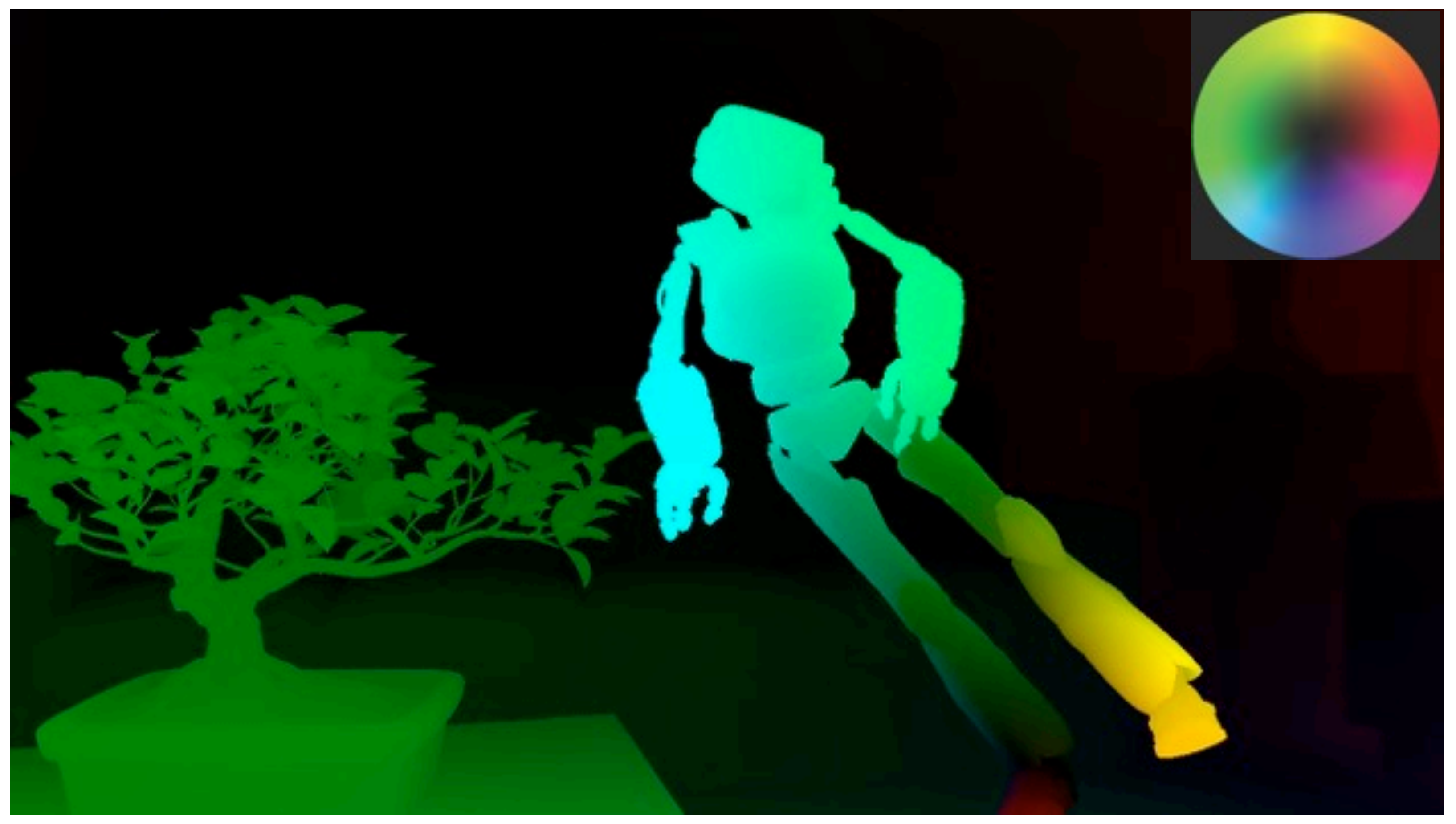
<click>
But we bootstrap the flow computation with the geometry motion vectors to better handle large displacements.

Irradiance Motion Estimation

Frame 1

Frame 2

1) Warp second frame by geometry flow

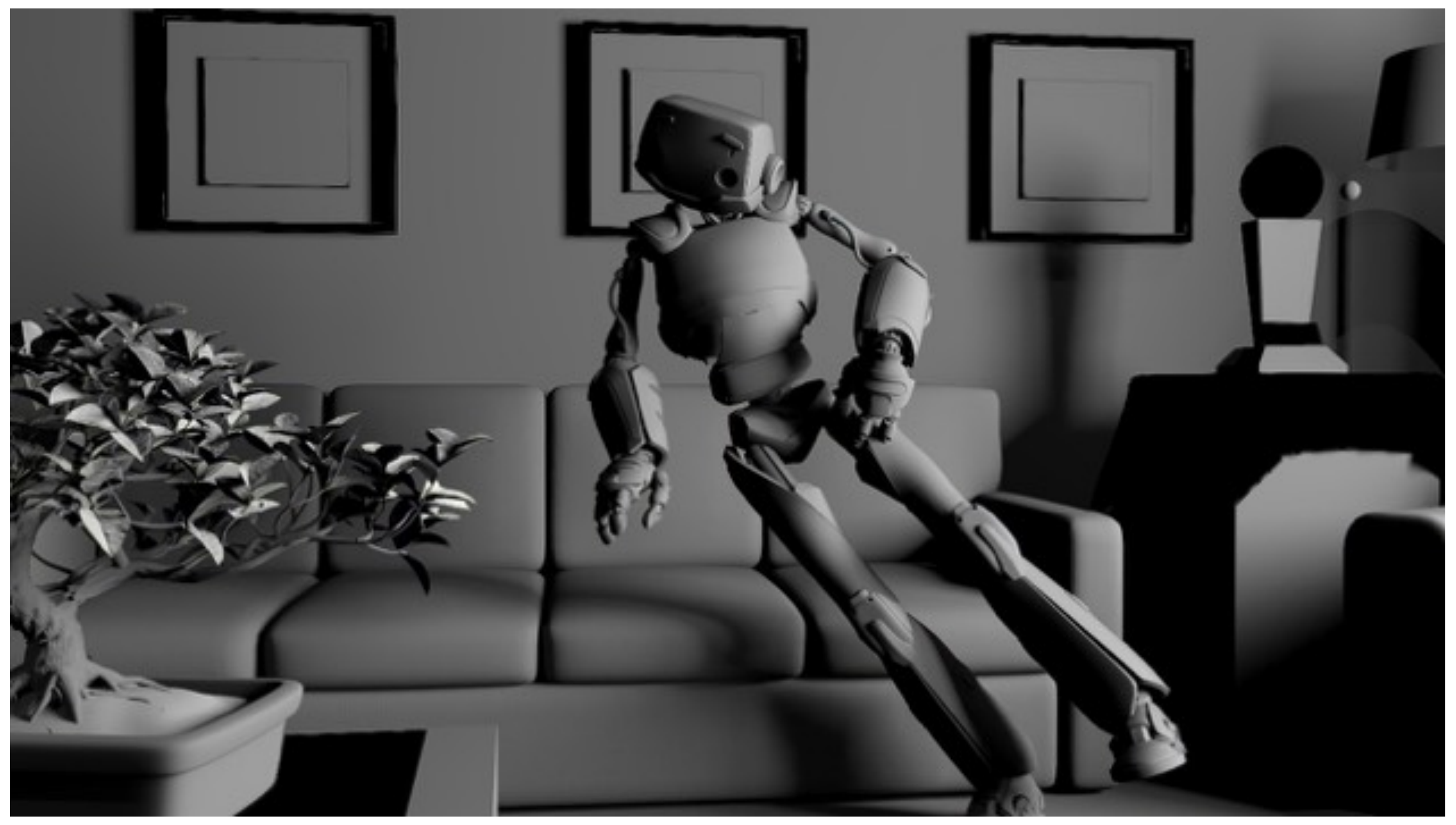


Geometry motion vectors

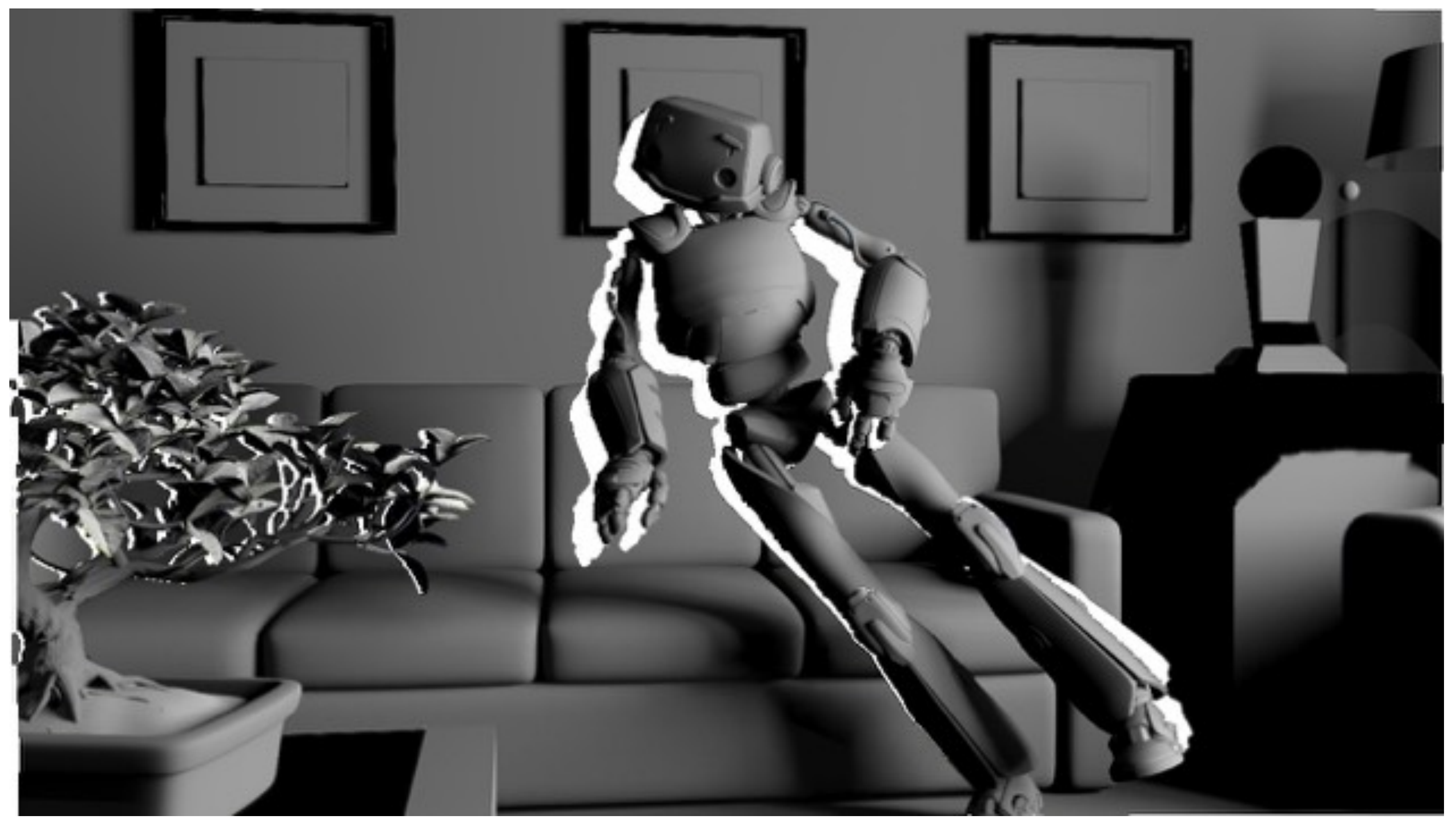
This bootstrapping comes down to first warping the second frame using the geometry motion vectors ...

Irradiance Motion Estimation

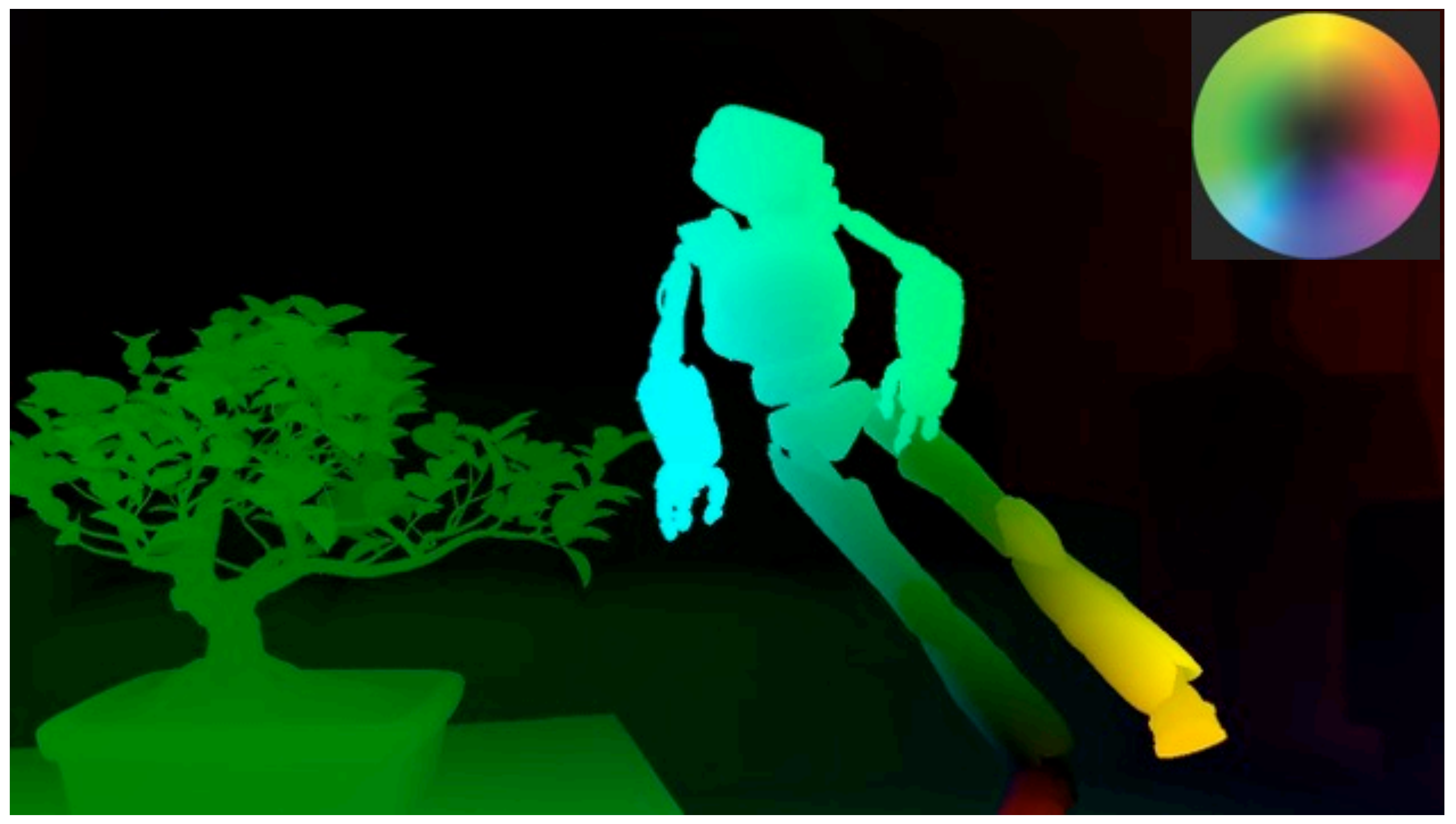
Frame 1



Frame 2



1) Warp second frame by geometry flow



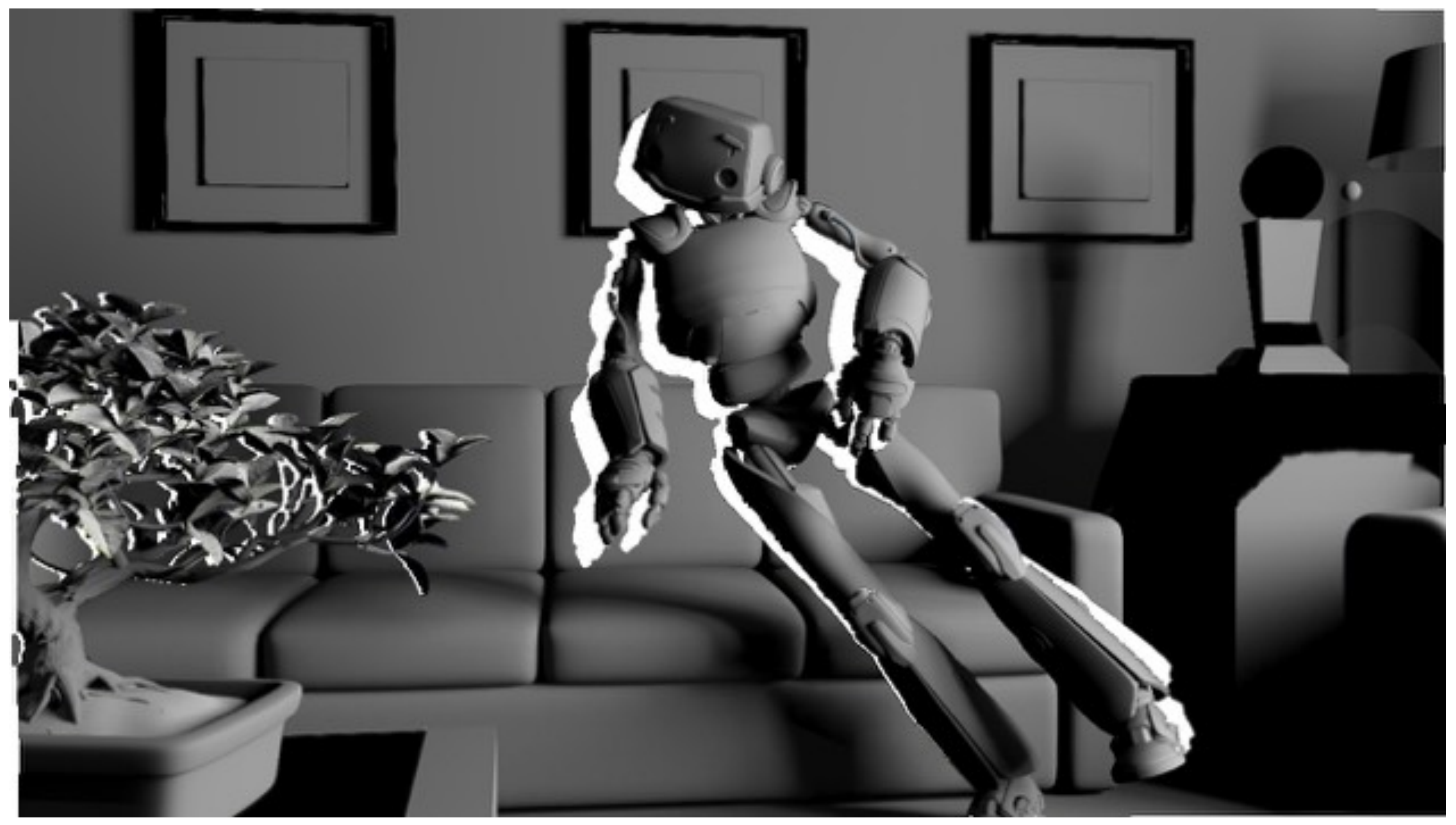
Geometry motion vectors

... which aligns all scene objects, and the only remaining apparent motion is due to the moving shadows.

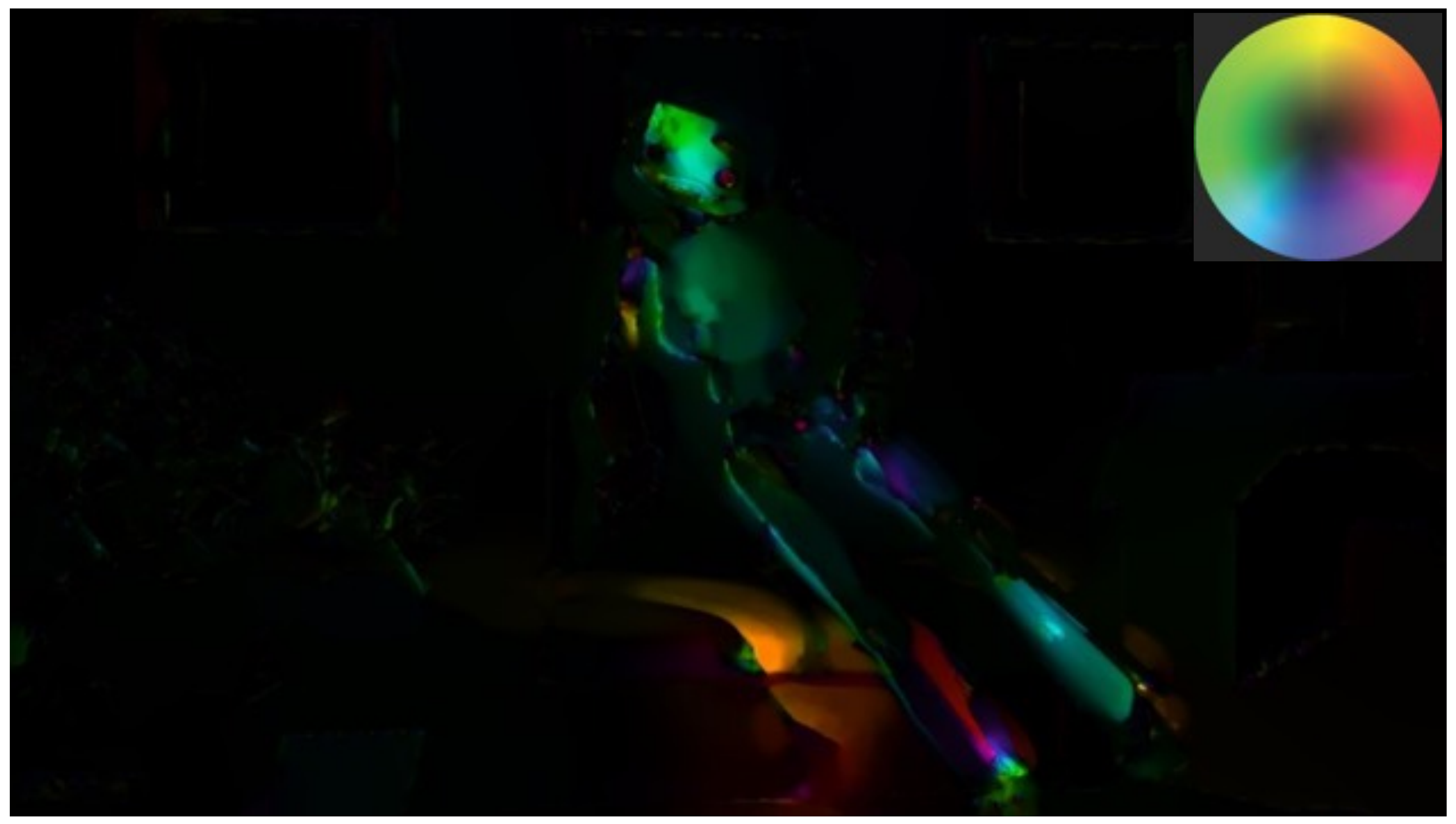
Irradiance Motion Estimation

Frame 1

Frame 2



2) Compute residual motion vectors using optical flow



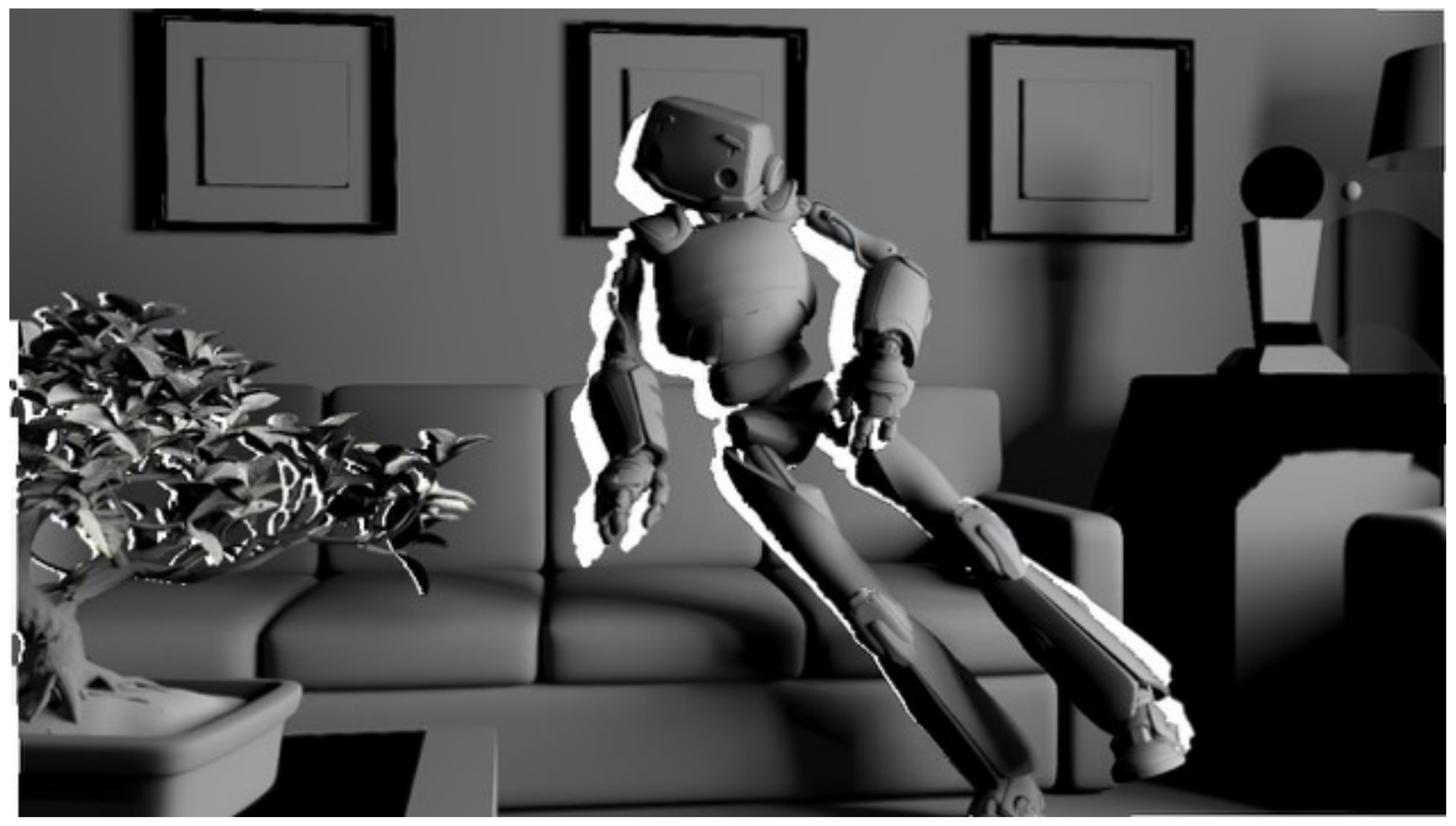
Residual motion vectors

Then we can use optical flow to estimate this apparent, residual motion ...

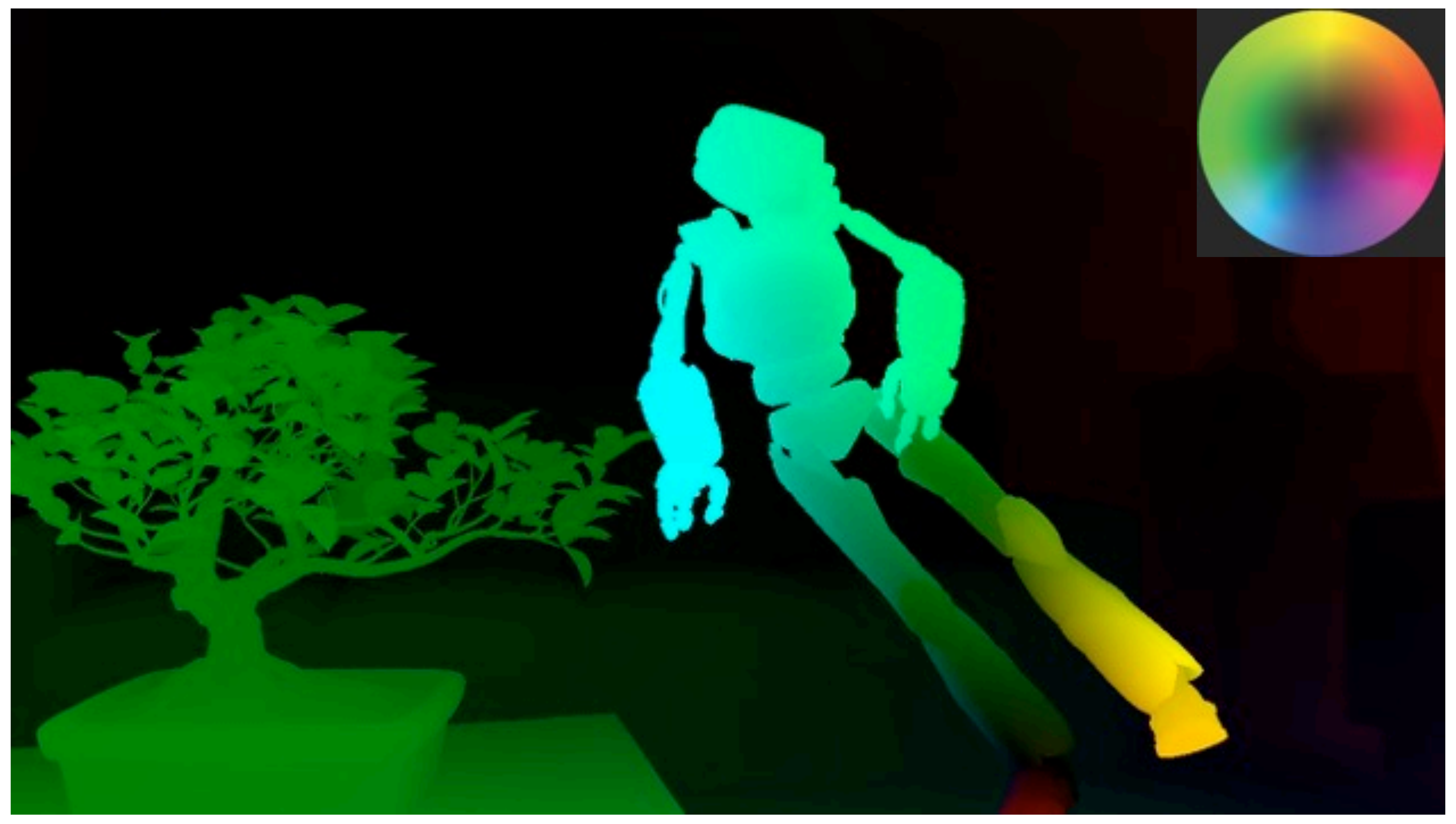
Irradiance Motion Estimation

Frame 1

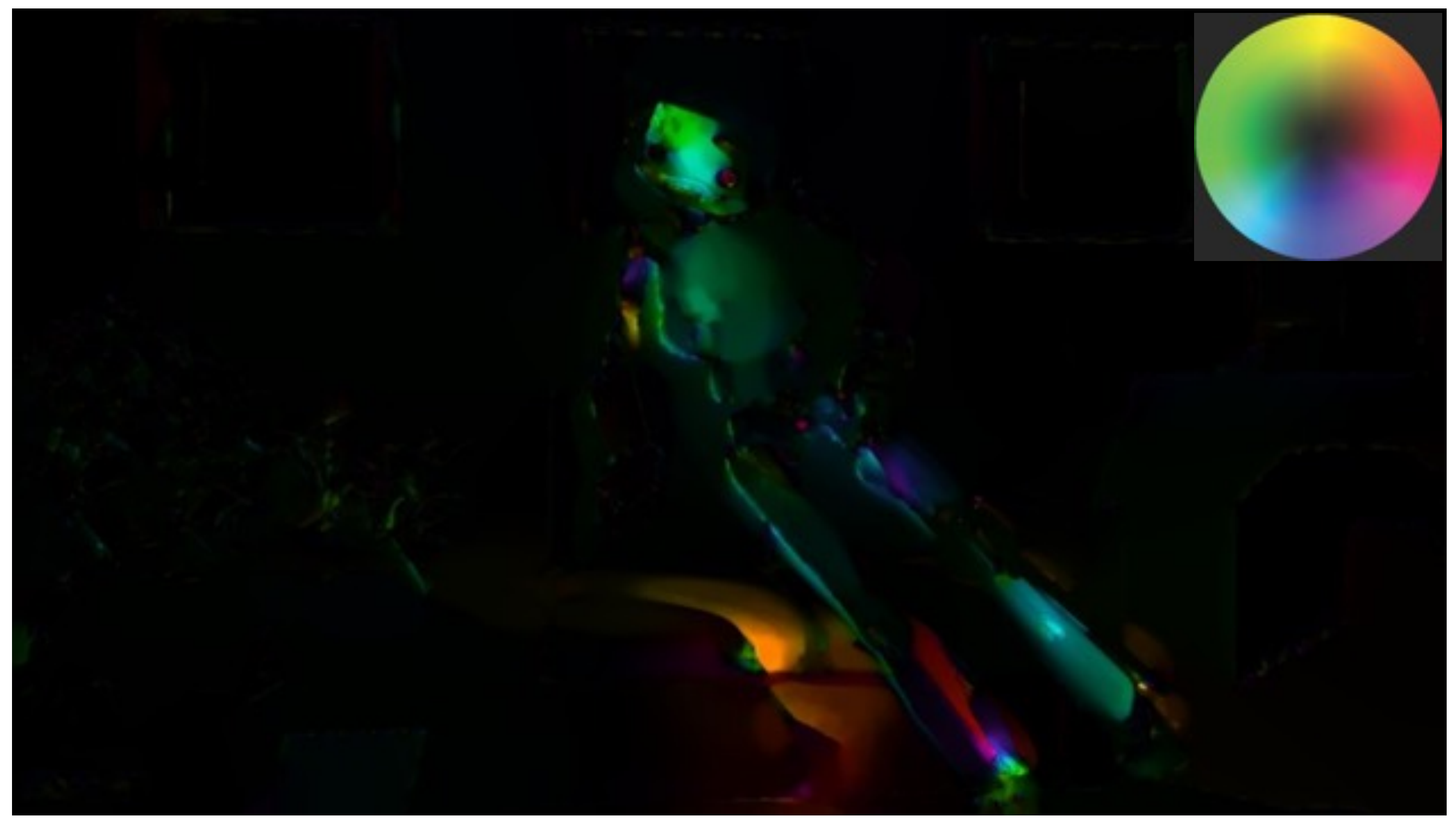
Frame 2



3) Add residual to geometry motion vectors



+

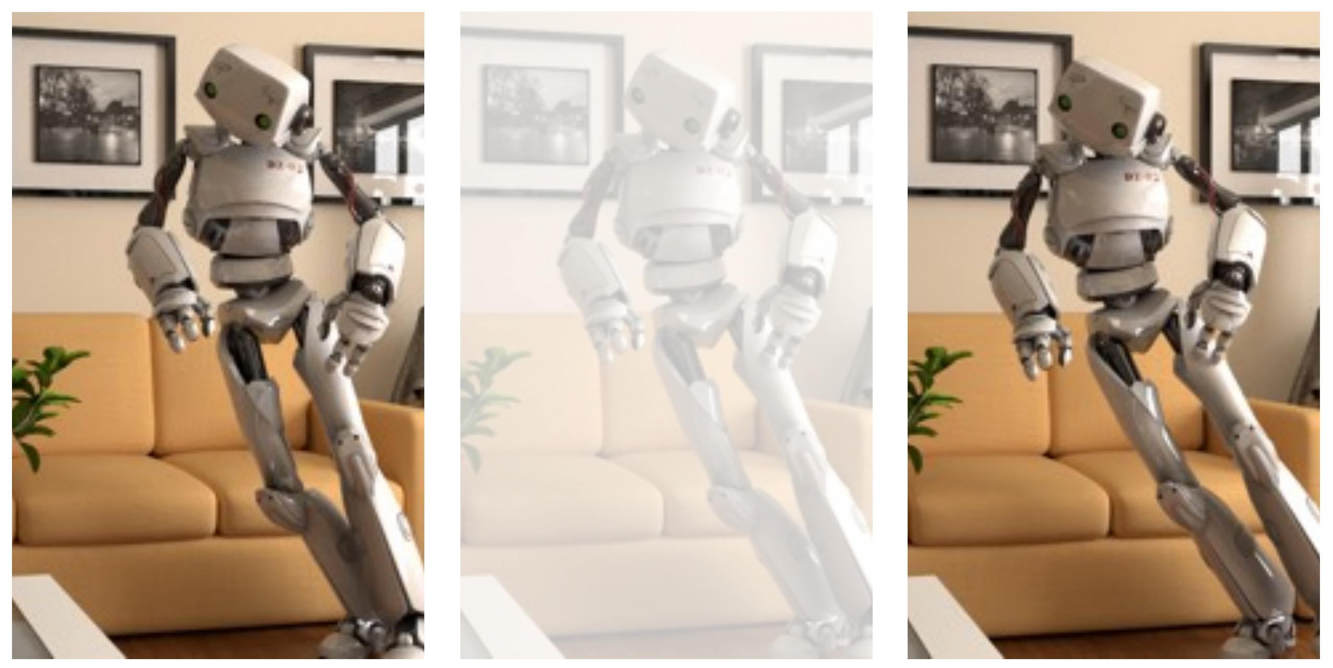


Geometry motion vectors

Residual motion vectors

... and add it to the geometry motion vectors to get the final motion estimate for the irradiance component.

Results


Temporal interpolation

Denoising

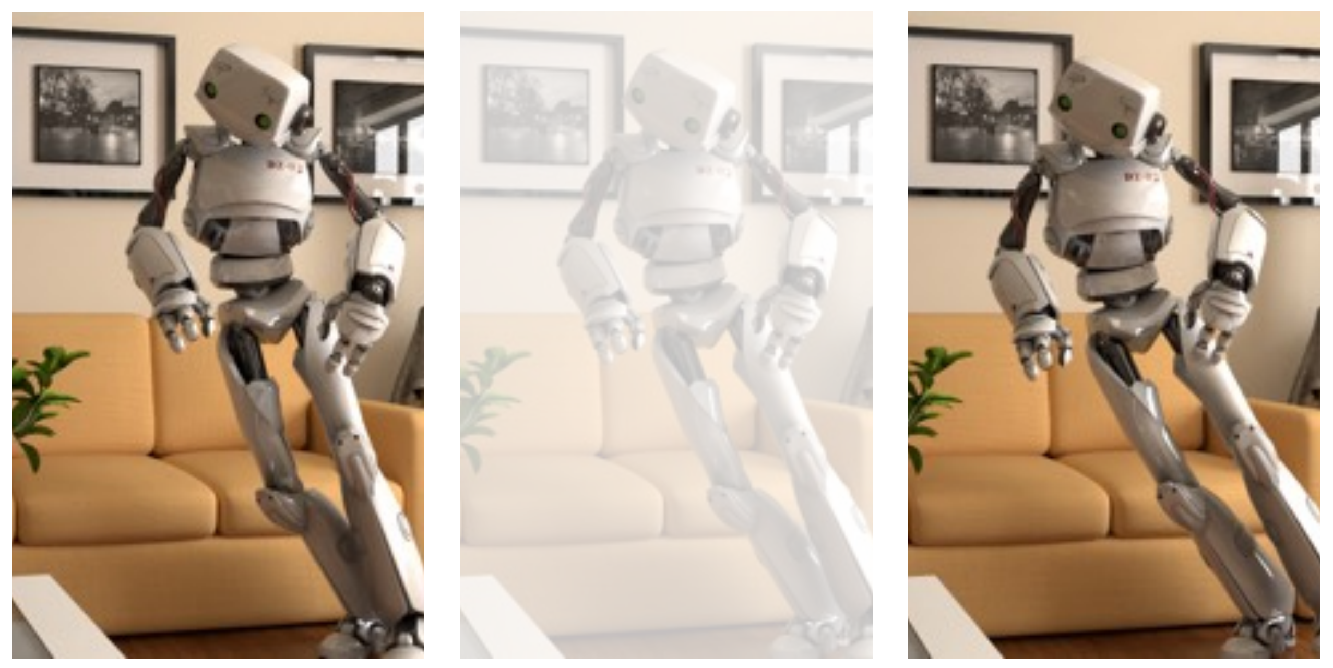
Spatial upsampling

Disney Research igl ETH zürich Walt Disney Animation Studios

We now demonstrate the benefits of our decomposition for three different post-processing applications.

<click>
And we start with temporal interpolation

Results



Temporal interpolation

Denoising

Spatial upsampling

We now demonstrate the benefits of our decomposition for three different post-processing applications.

<click>
And we start with temporal interpolation

Temporal Interpolation

Keyframe 1

Keyframe 2

Here, the idea is to only render a few keyframes, for instance every 4th frame ...

Temporal Interpolation

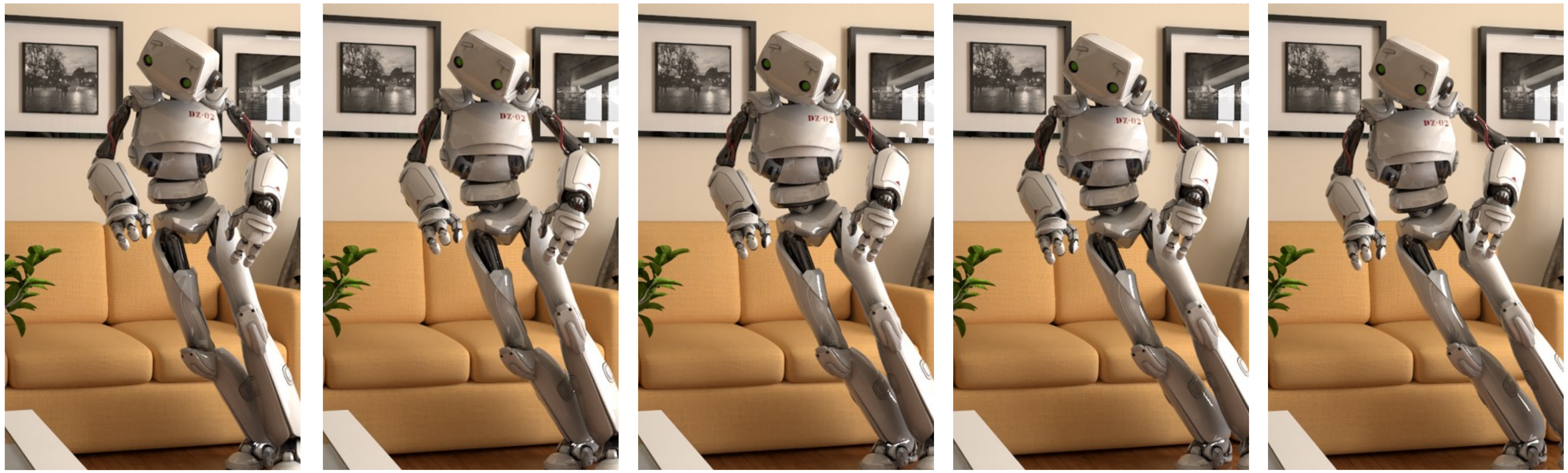
Keyframe 1

Keyframe 2

... and then reconstruct the entire sequence from these keyframes.

This is typically done by splatting pixels from the keyframes to the in-between frames according to computed motion vectors.

Temporal Interpolation



... and then reconstruct the entire sequence from these keyframes.

This is typically done by splatting pixels from the keyframes to the in-between frames according to computed motion vectors.

Specular Interpolation

Naive interpolation
(final color + geometry motion vectors)

Our result
(decomposition + per-component motion vectors)

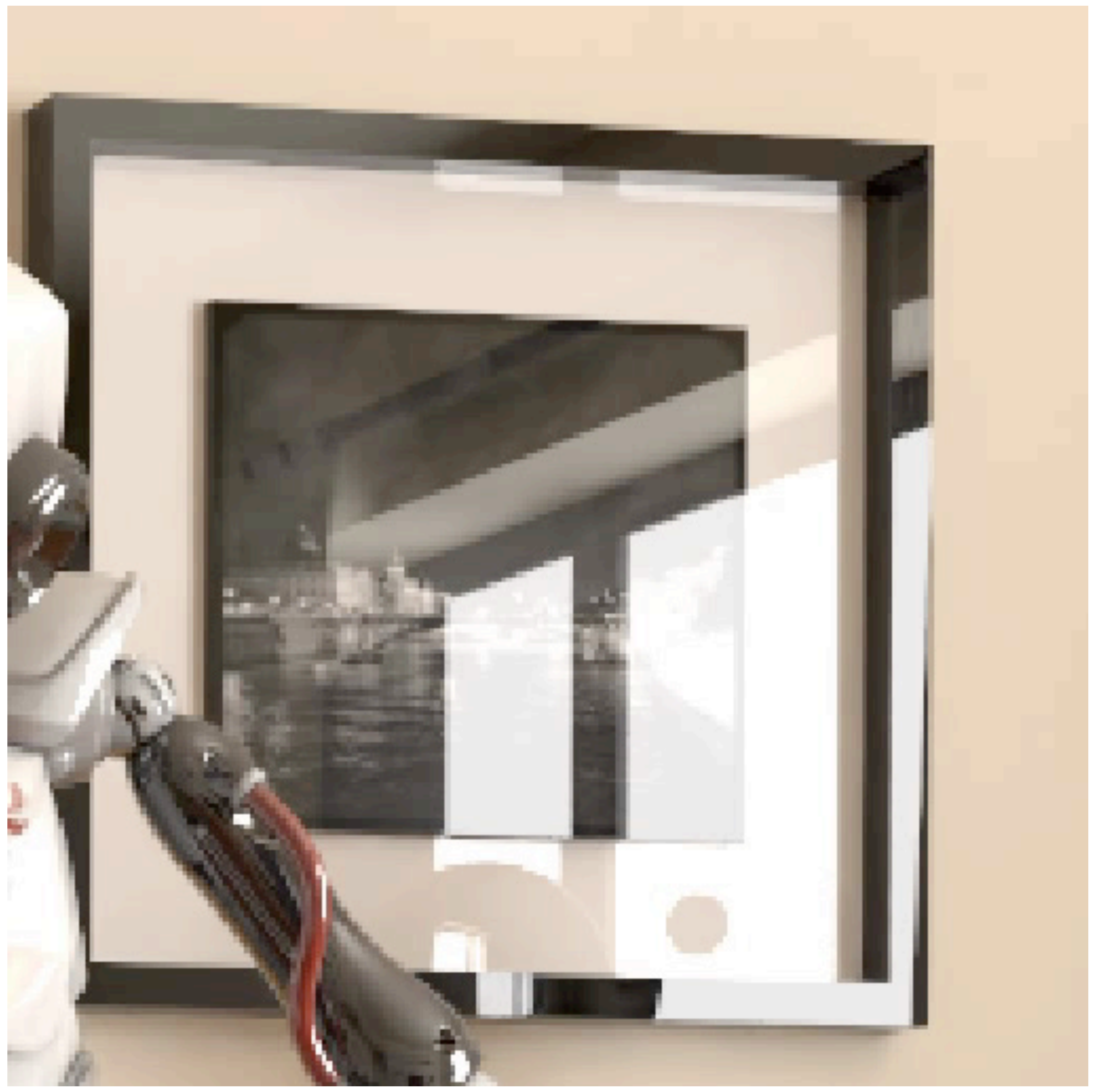
Here on the left we see the result of a naive interpolation approach that splats final pixels colors using the geometry motion vectors.

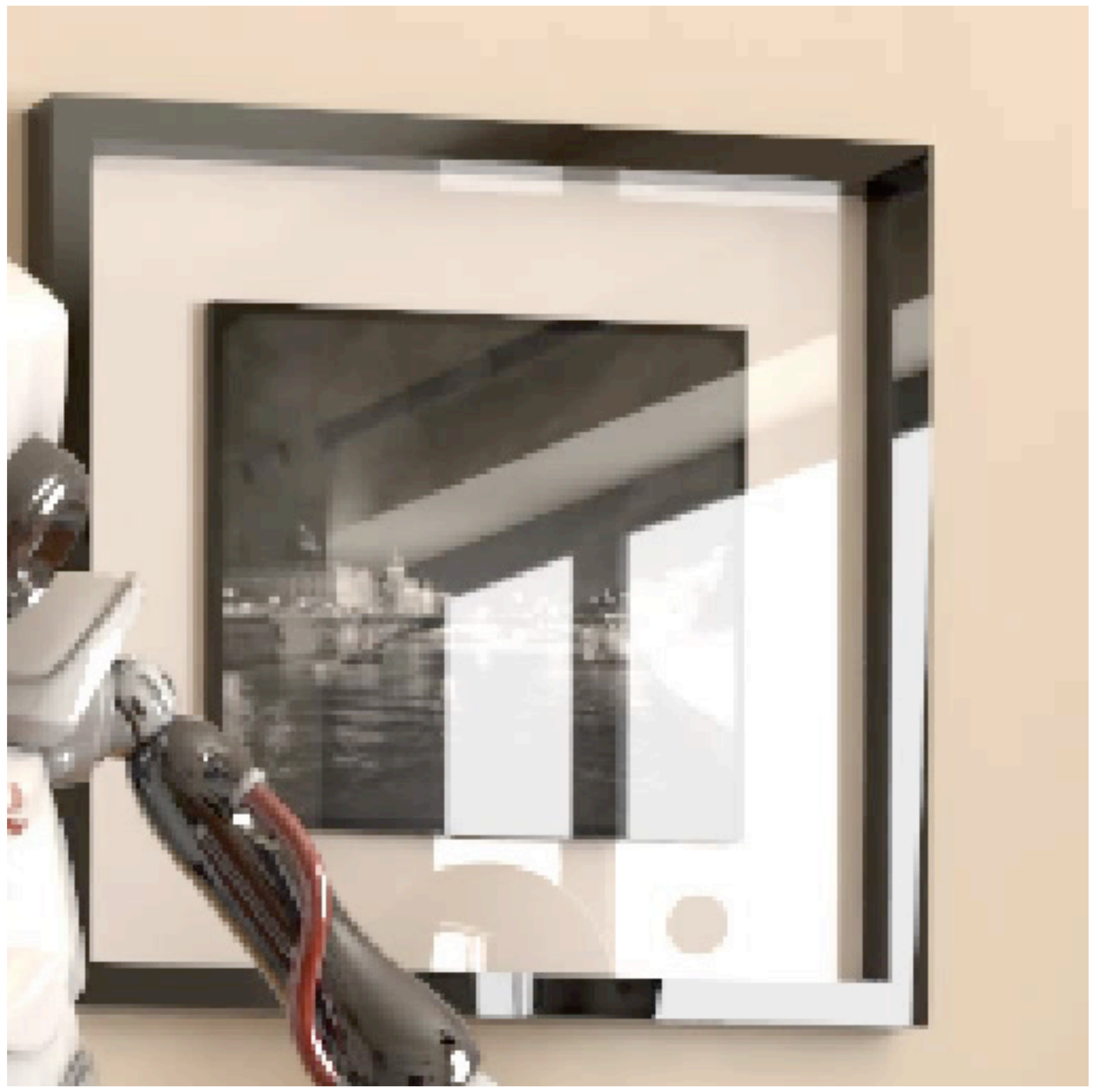
As the latter do not capture the motion of the specular reflections, disturbing ghosting artifacts appear.

Our result on the right correctly handles specular reflections, as we interpolate each component separately using the corresponding motion vectors.

Consequently, this result verifies the accuracy of the specular motion vectors, computed using temporal manifold exploration.

Specular Interpolation


Naive interpolation
(final color + geometry motion vectors)


Our result
(decomposition + per-component motion vectors)

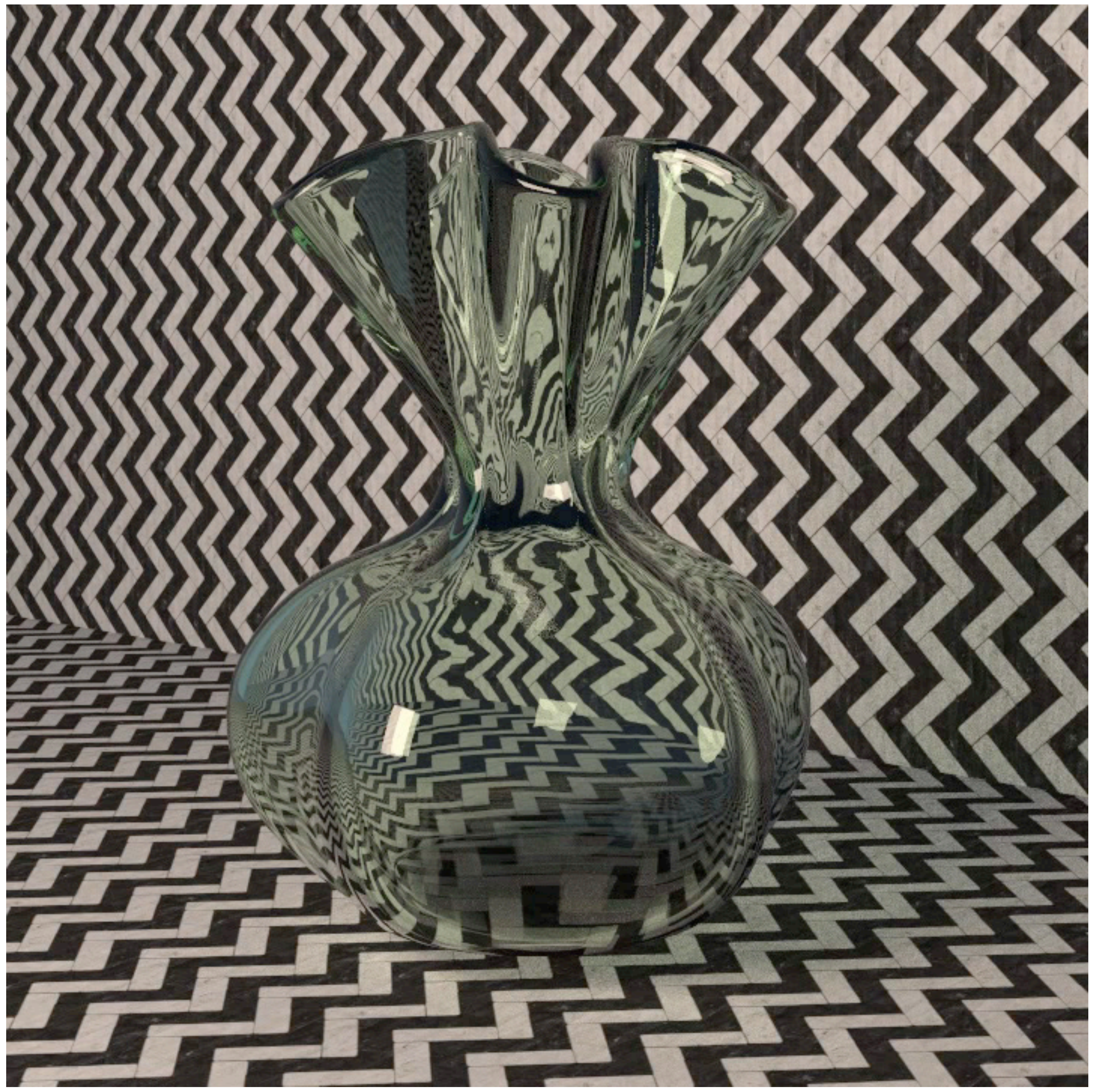
Here on the left we see the result of a naive interpolation approach that splats final pixels colors using the geometry motion vectors.

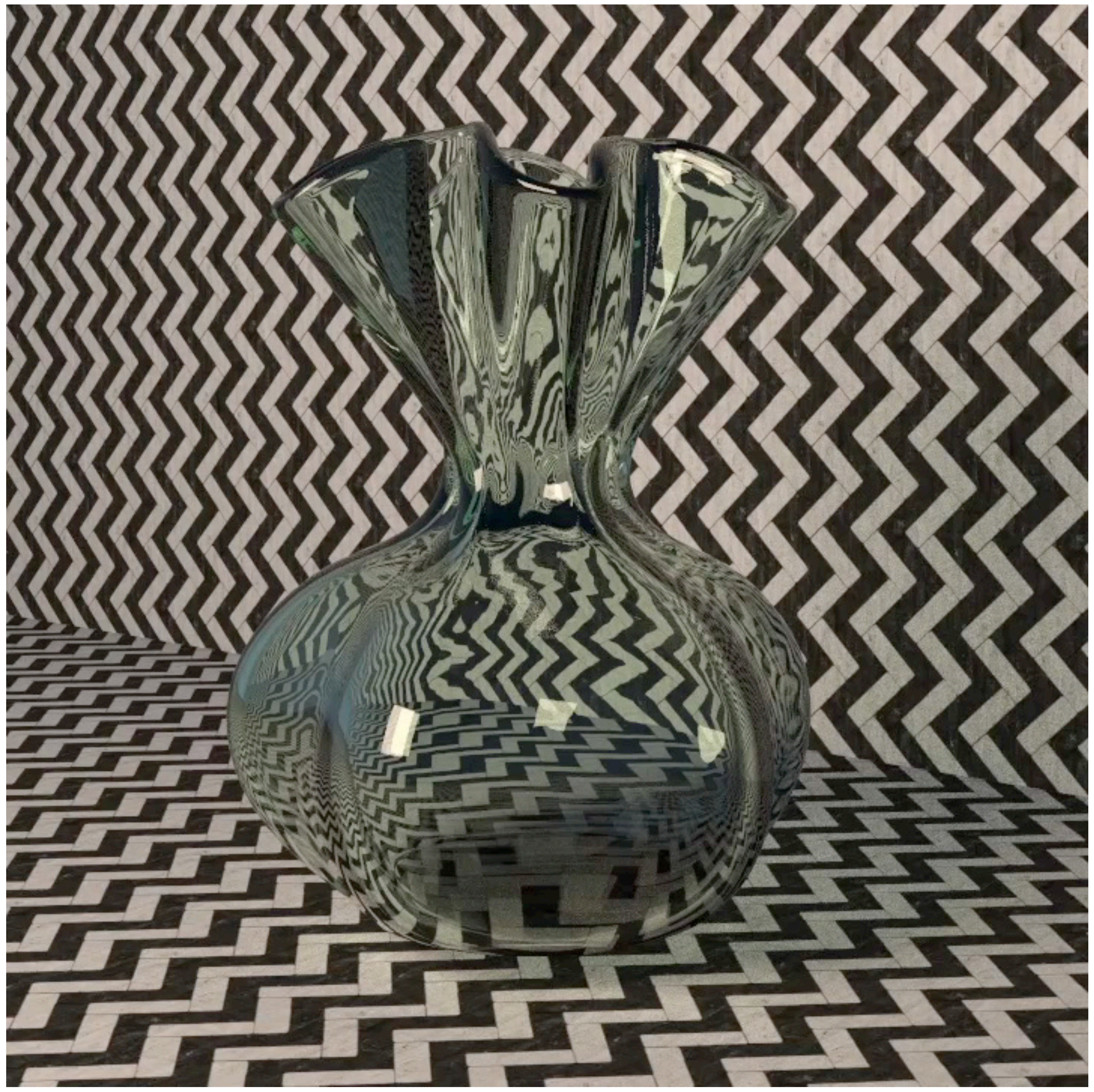
As the latter do not capture the motion of the specular reflections, disturbing ghosting artifacts appear.

Our result on the right correctly handles specular reflections, as we interpolate each component separately using the corresponding motion vectors.

Consequently, this result verifies the accuracy of the specular motion vectors, computed using temporal manifold exploration.

Complicated Specular Interpolation


Naive interpolation

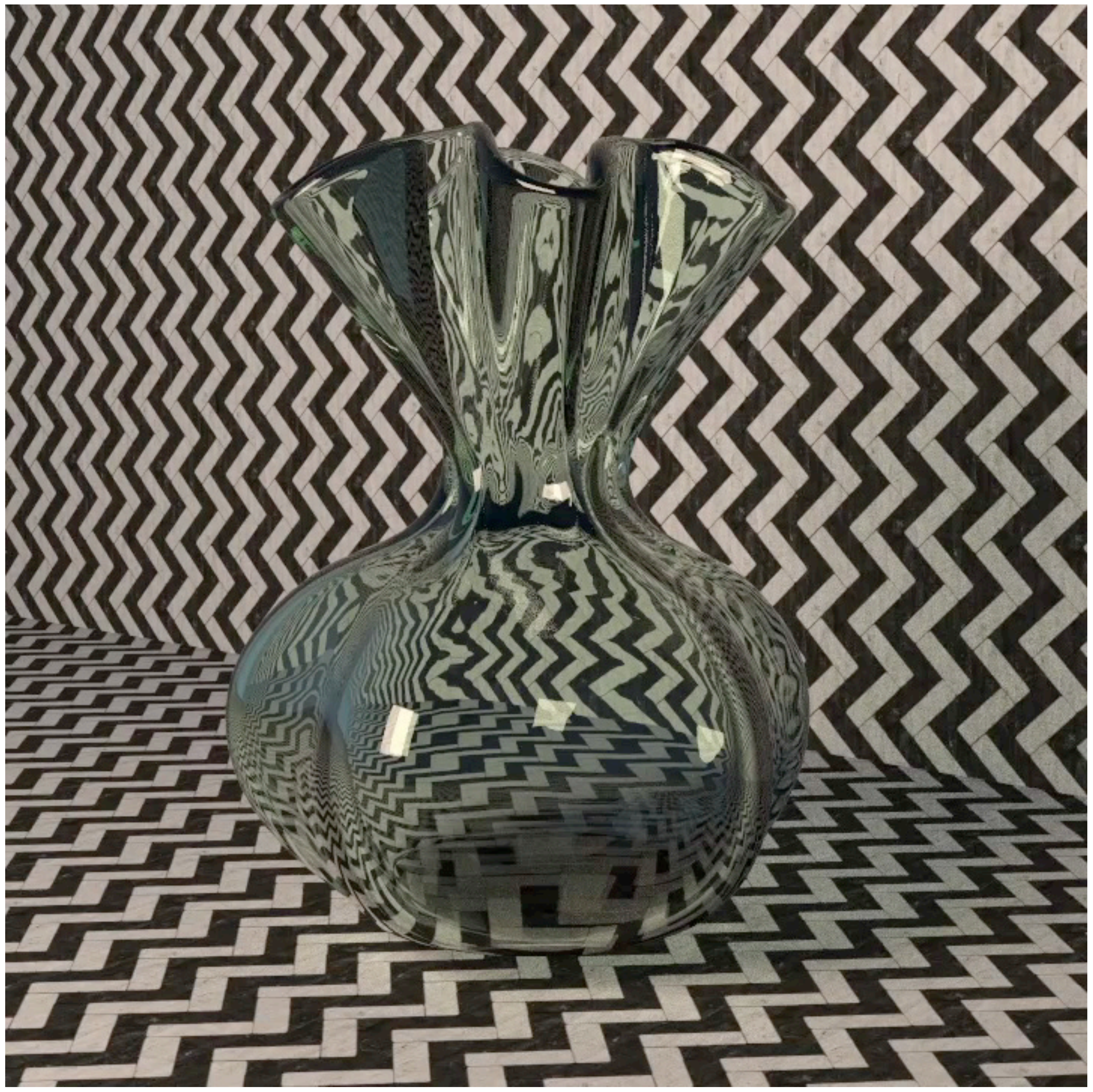

Our result (adding **ETTTD.*** component)

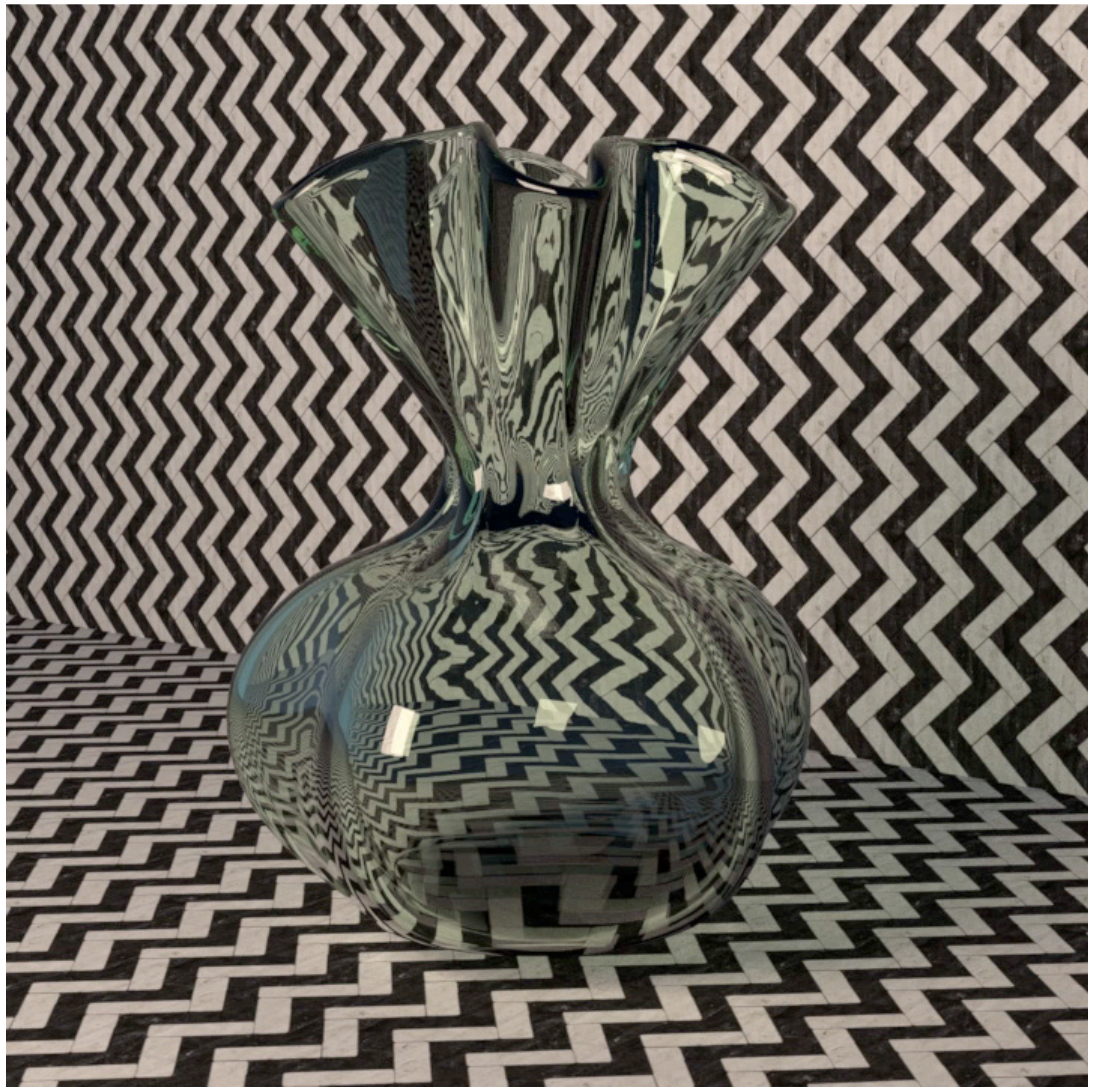
This example features intricate specular transport.
A naive interpolation shown on the left exhibits very strong artifacts, especially for the textured background seen through the vase.

We can handle this challenging case by leveraging the flexibility of our decomposition and adding another component to our decomposition to capture fourfold specular transmission paths.

We again estimate motion vectors for this component using temporal manifold exploration, which allows us to obtain favorable interpolation results ....

Complicated Specular Interpolation


Our result (adding **ETTTD.*** component)

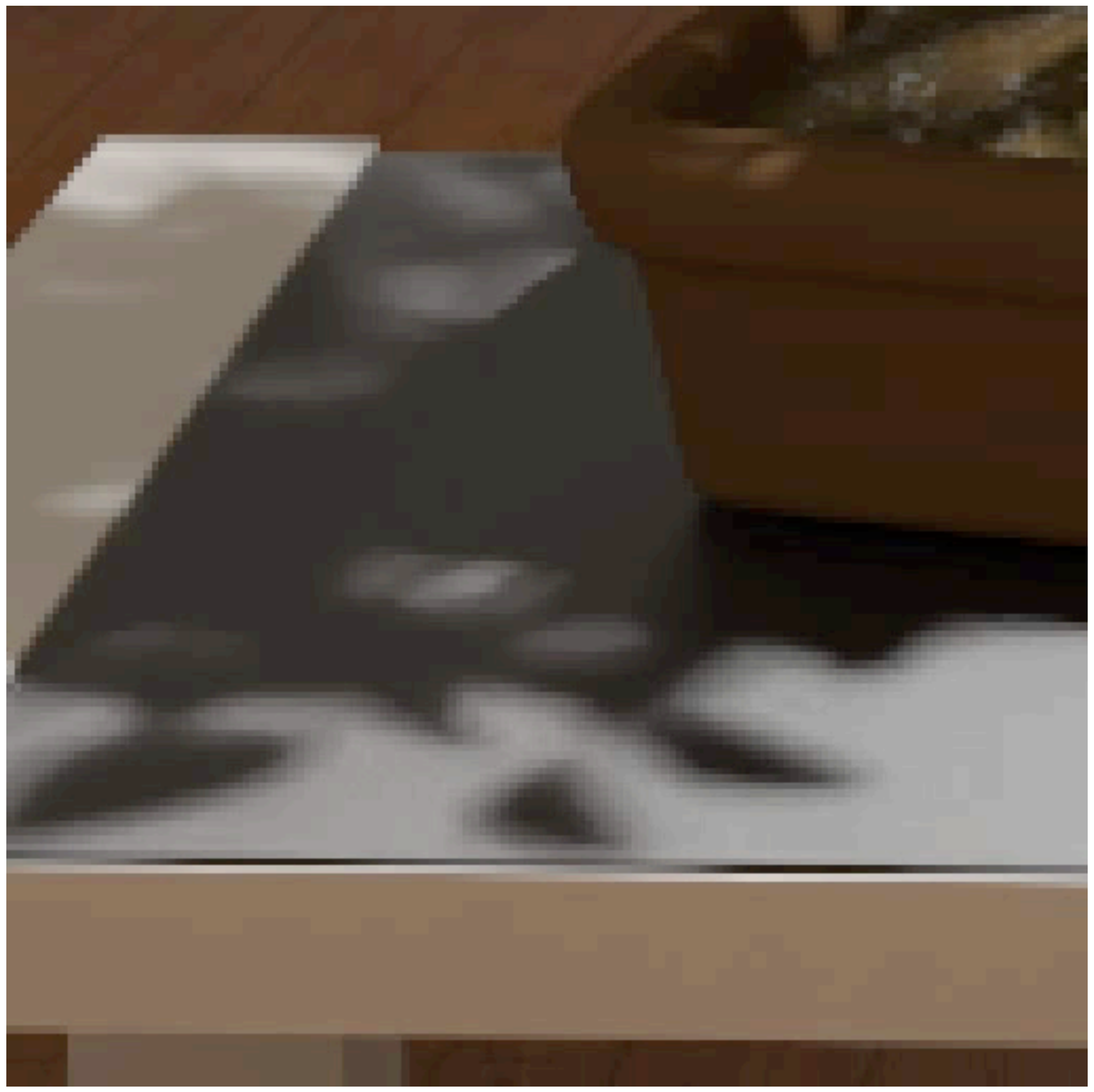

Ground truth

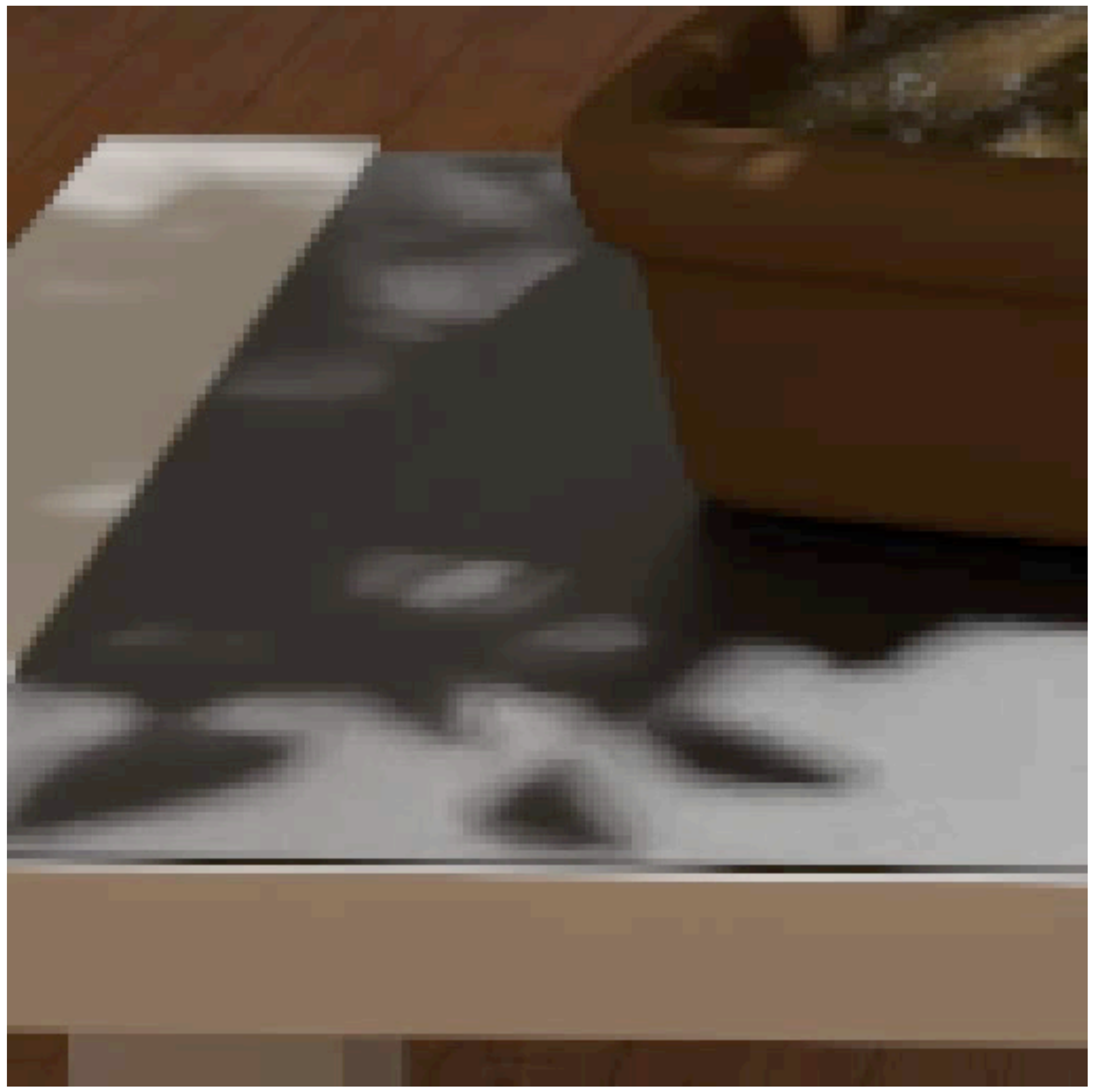
Disney Research igl ETH zürich Walt Disney Animation Studios

Monday, June 29, 15

... that also closely resemble the ground truth shown on the right.

Shadow Interpolation Results


Naive interpolation


Our result

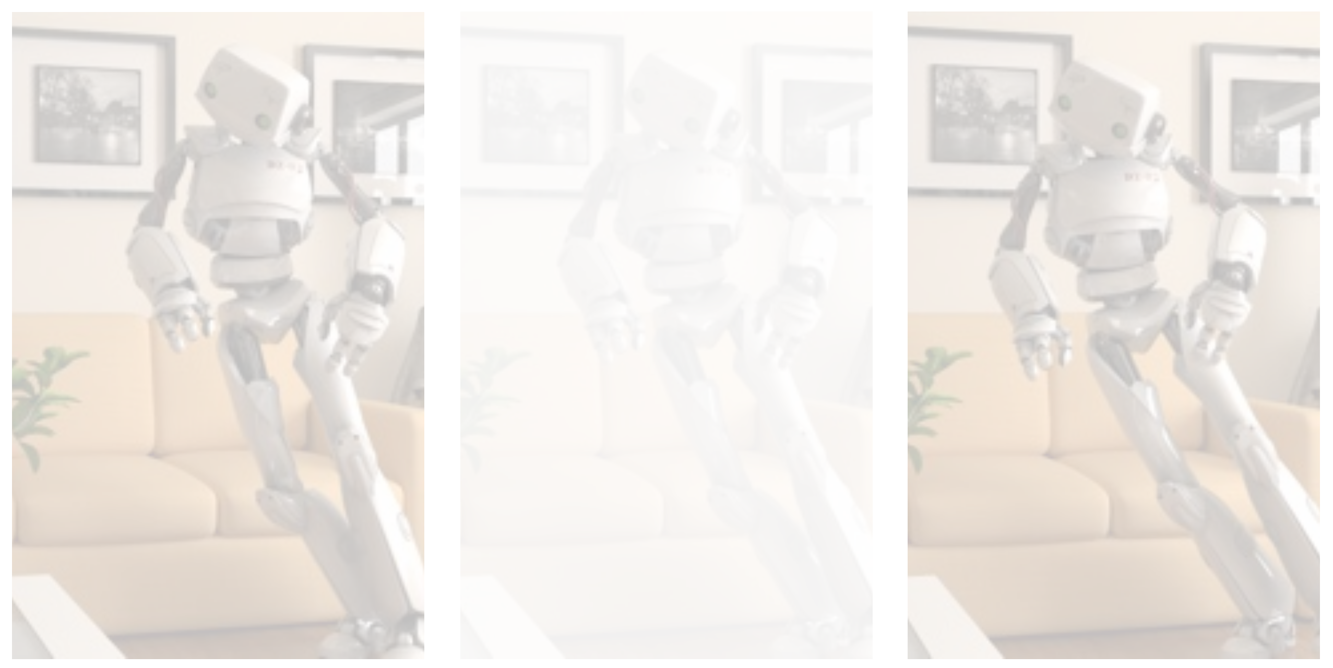
Disney Research igl ETH zürich Walt Disney Animation Studios

This last example illustrates the handling of moving shadows.

On the left we show that a naive interpolation yields ghosting at the moving shadows.

Our result on the right remedies these artifacts as we separately interpolate the irradiance components using the motion vectors estimated using optical flow.

Results



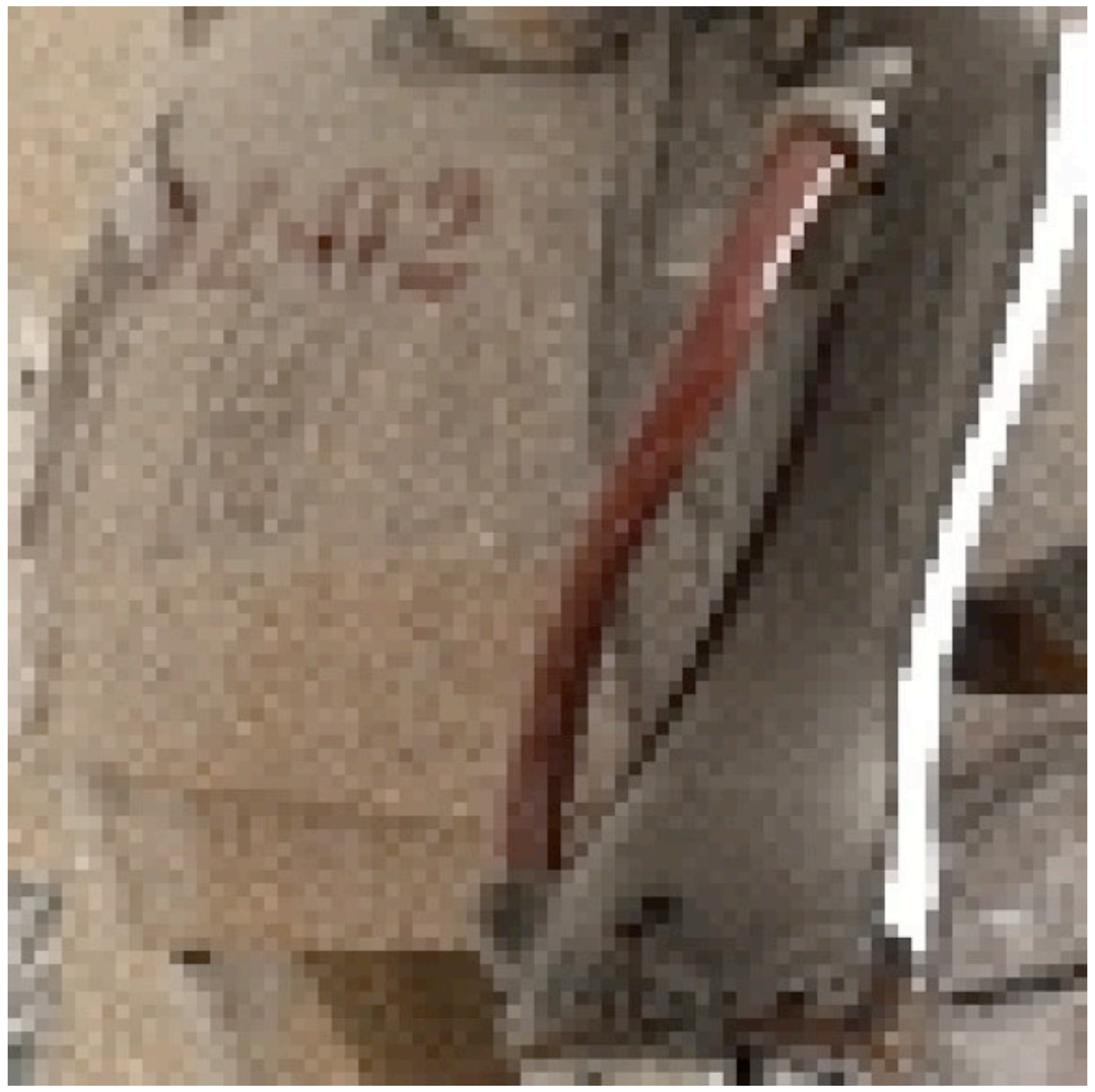
Temporal interpolation

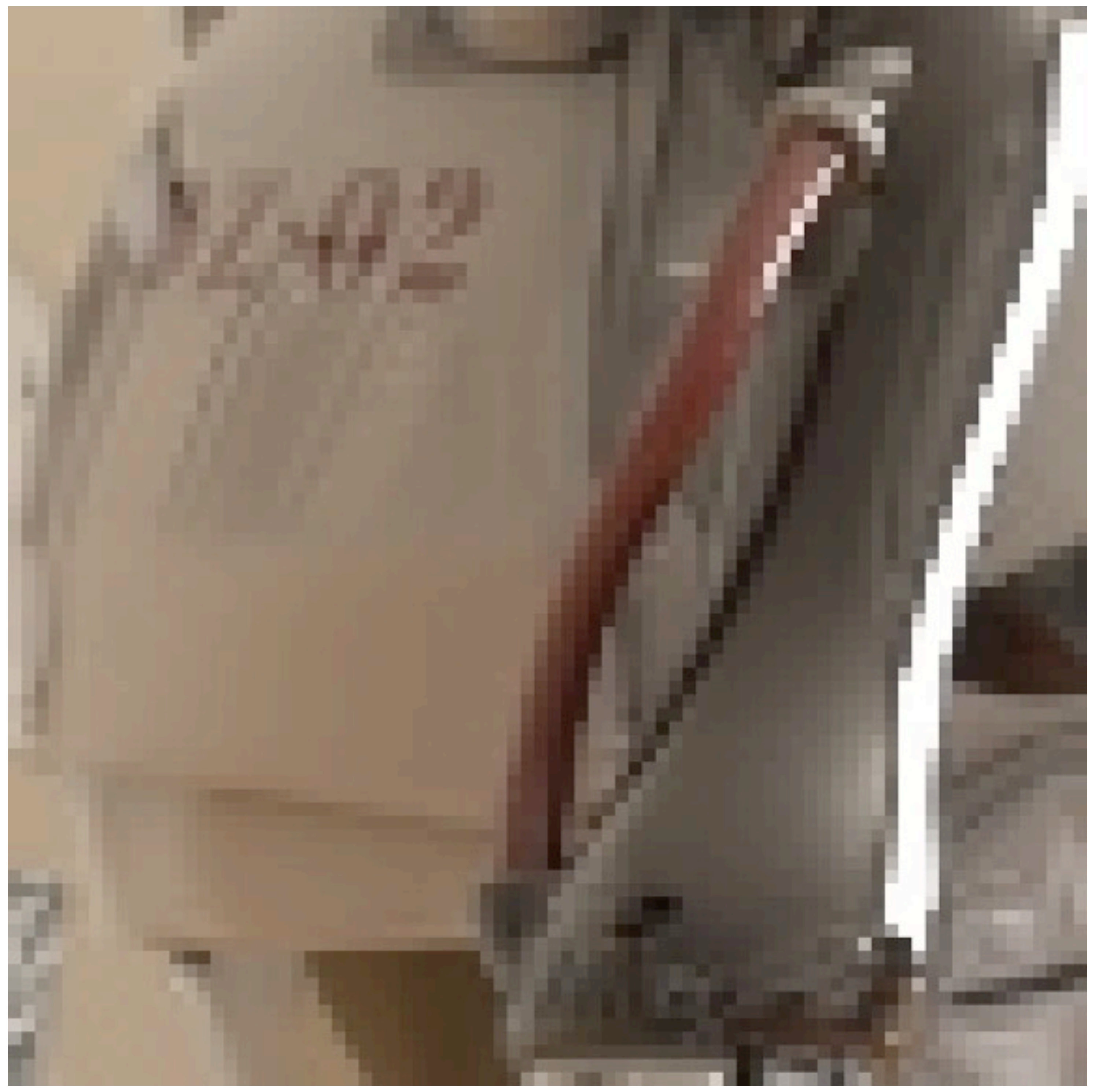
Denoising

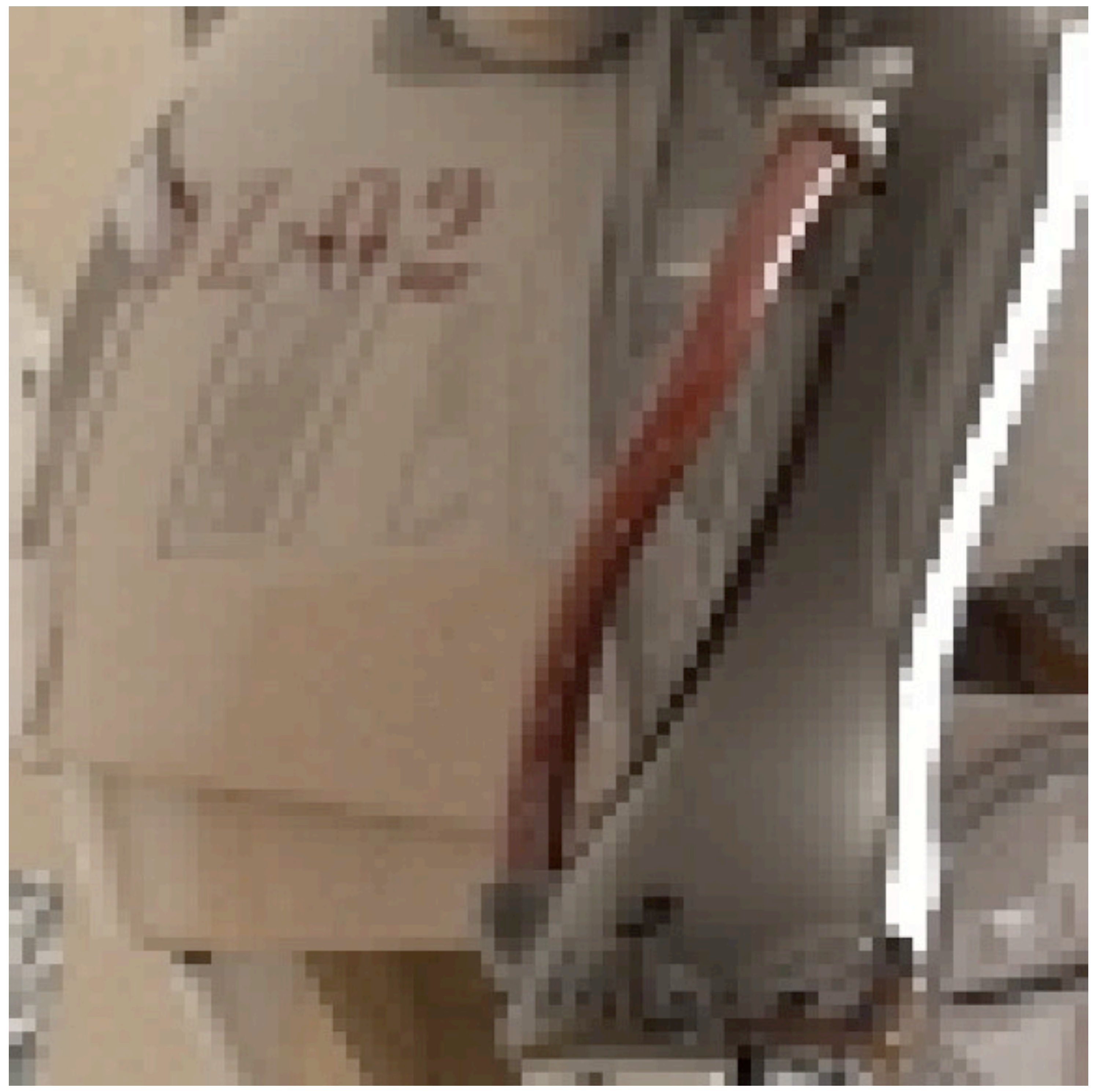
Spatial upsampling

We now move to denoising results

Denoising Results


Input (512 spp)


[Rousselle et al. 2013]

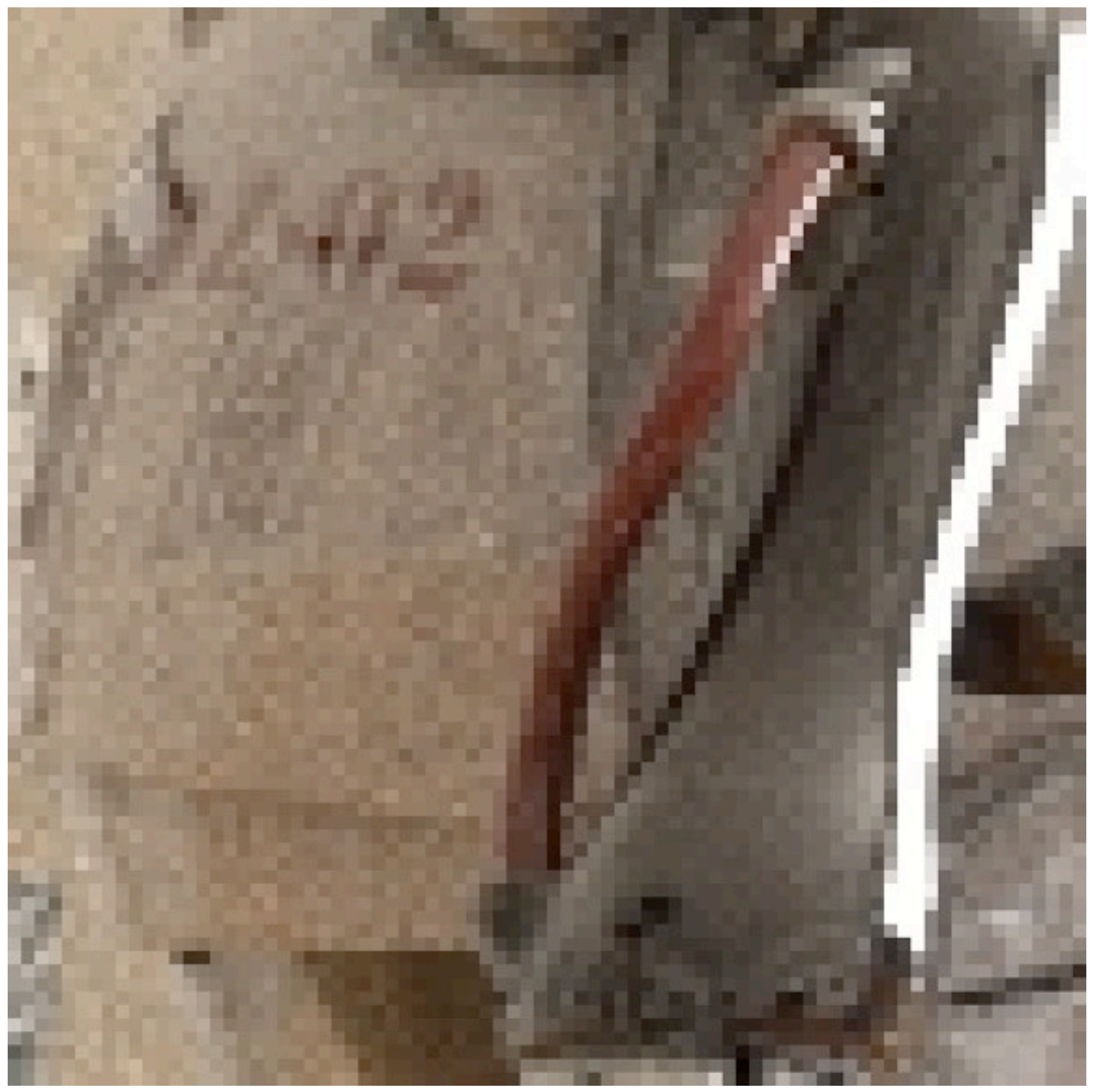

Ground truth (16k spp)

rMSE	$7.7 \cdot 10^{-3}$	$1.2 \cdot 10^{-3}$	

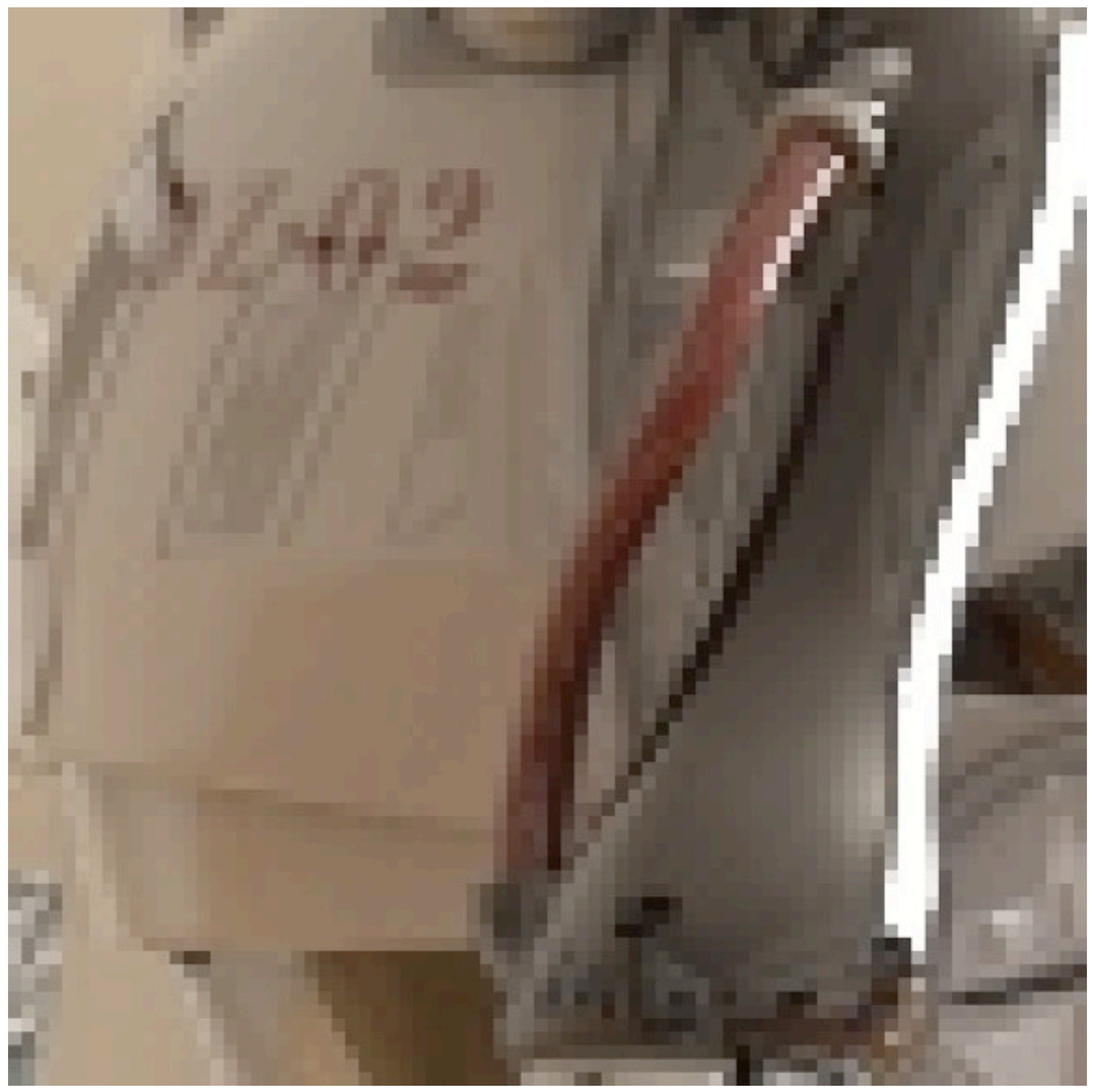
Let us first look at the results of a state-of-the-art denoising method shown in the middle.
It filters the final pixels colors using the feature buffers of the scene geometry, for example surface normals, as a guide.

While this successfully removes noise from the input on the left,
most details of the reflections are lost, which becomes apparent when comparing to the reference render on the right.

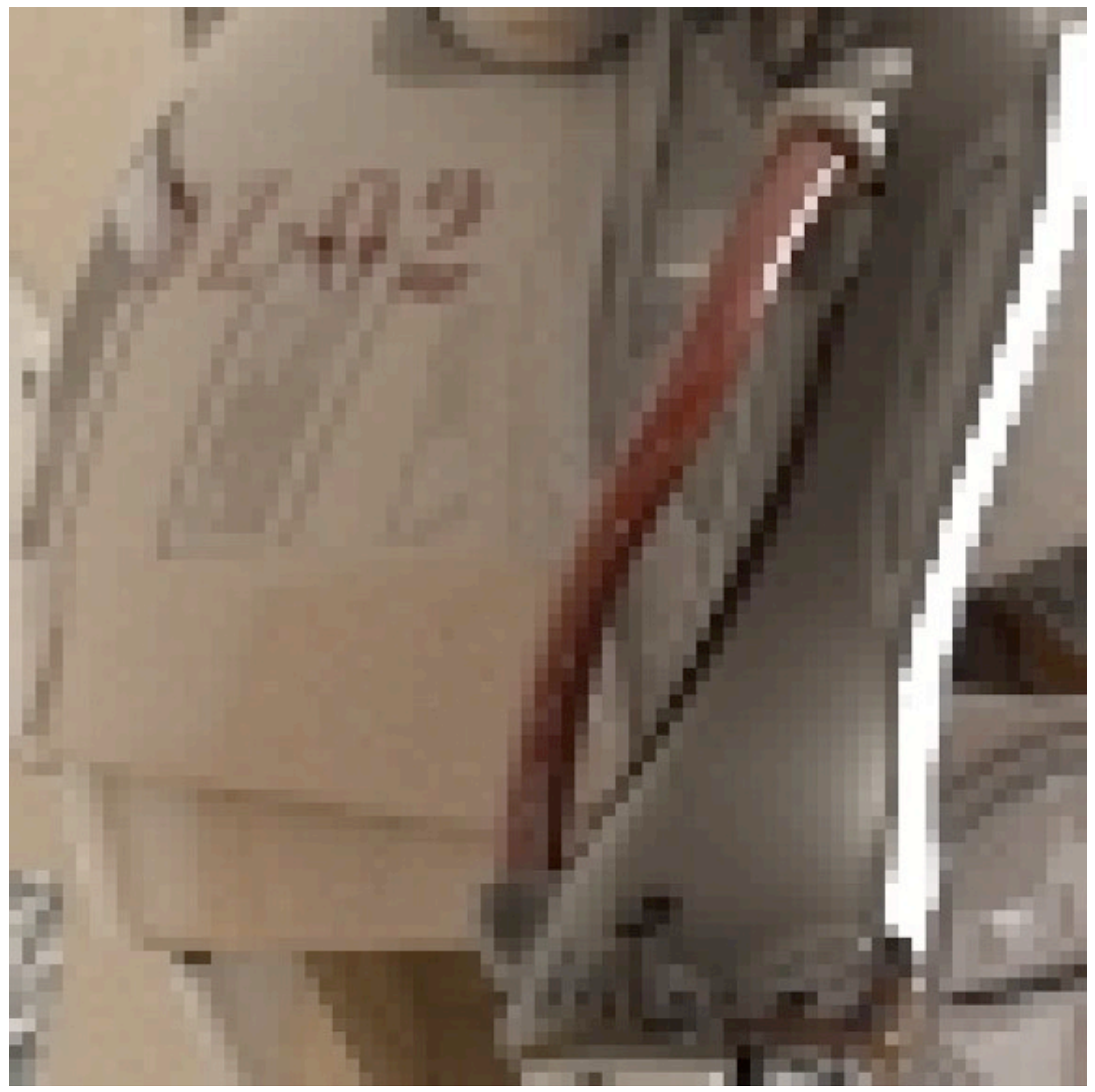
Denoising Results



Input (512 spp)



Our result



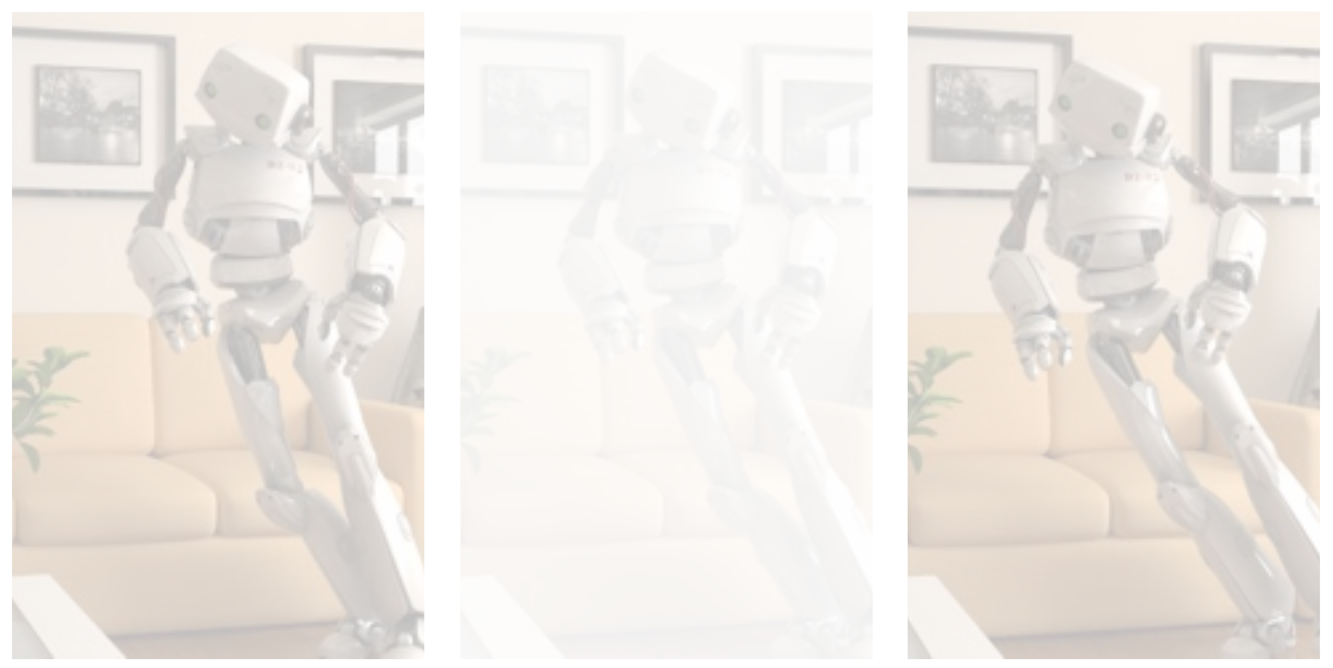
Ground truth (16k spp)

rMSE **$7.7 \cdot 10^{-3}$** **$0.8 \cdot 10^{-3}$**

Our result preserves details in the reflections as we denoise each component separately and we can leverage per-component feature buffers to guide the denoising.

This is reflected in a 30% decrease in relative MSE,
but should also be easily visible when comparing the results
<switch back and forth>

Results



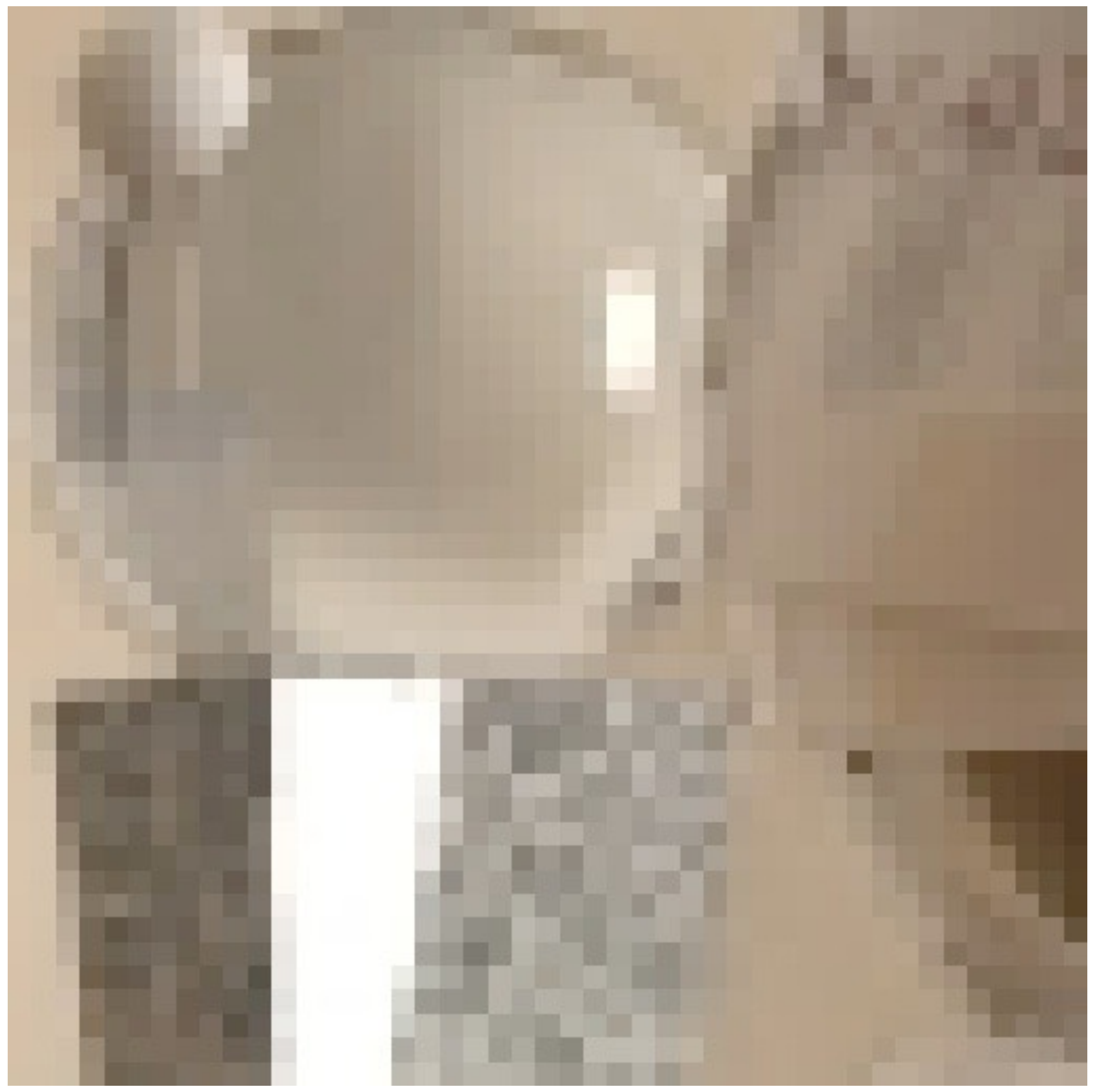
Temporal interpolation

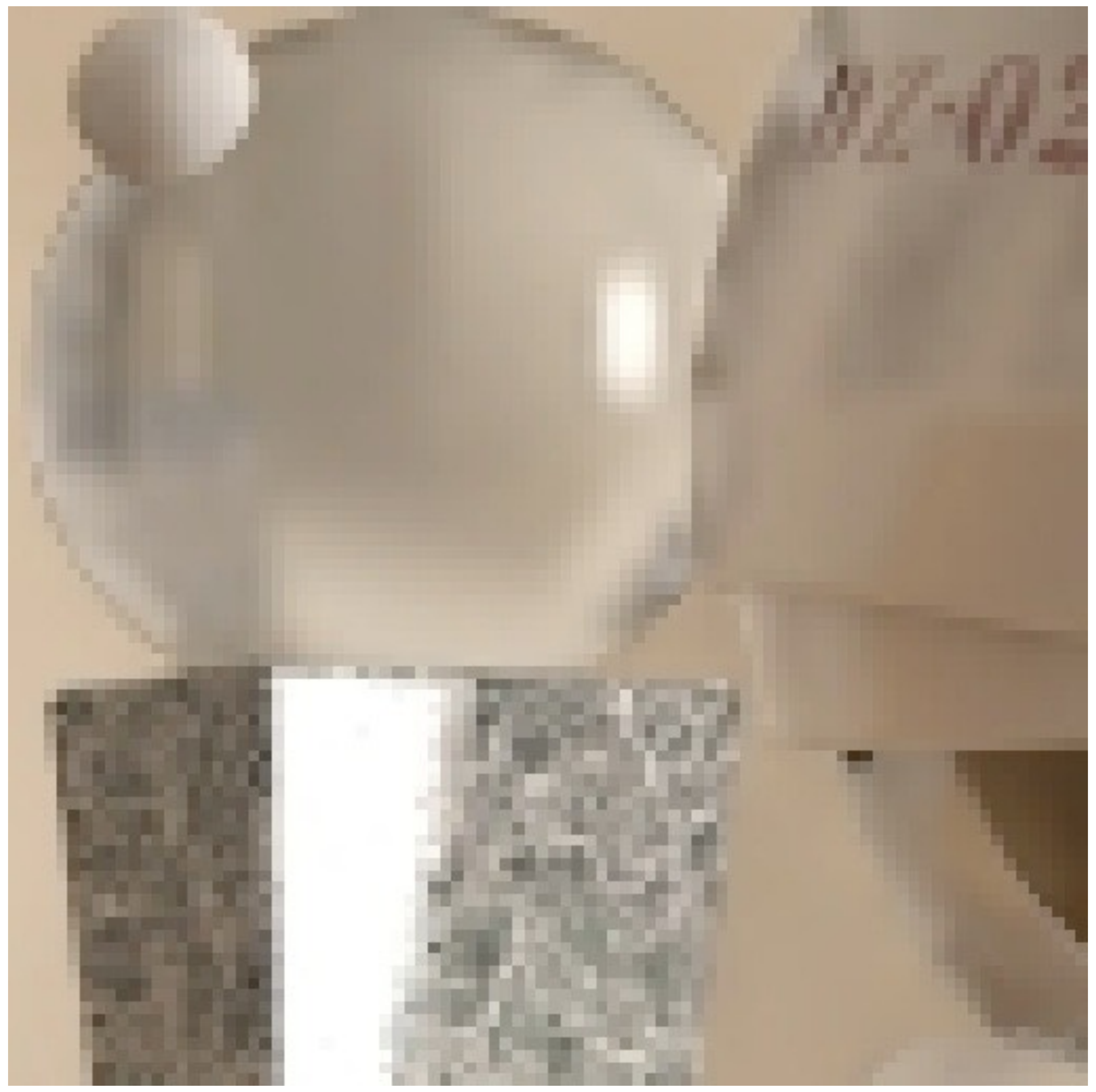
Denoising

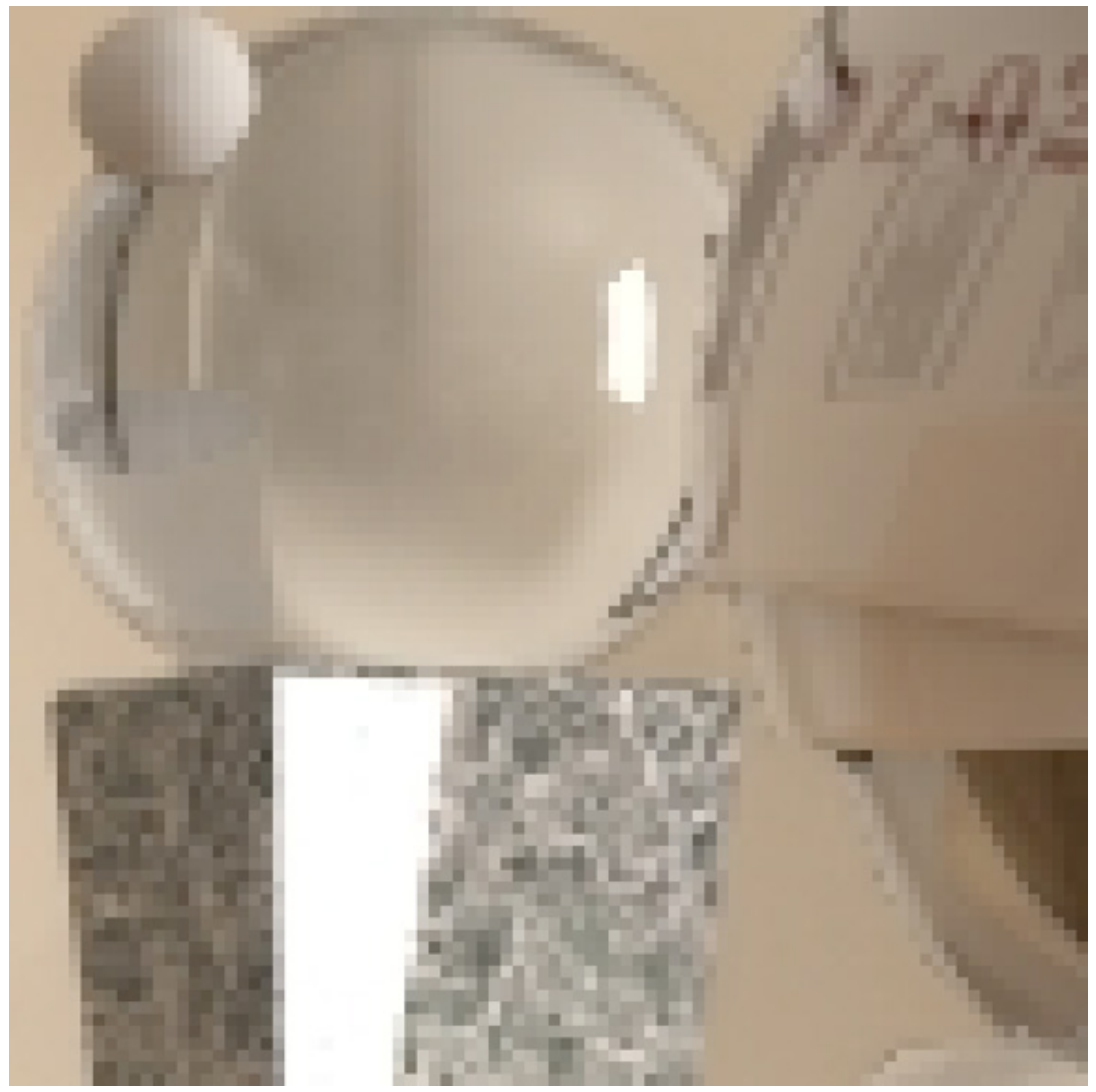
Spatial upsampling

Our last application is spatial upsampling

Spatial Upsampling


Low-res input

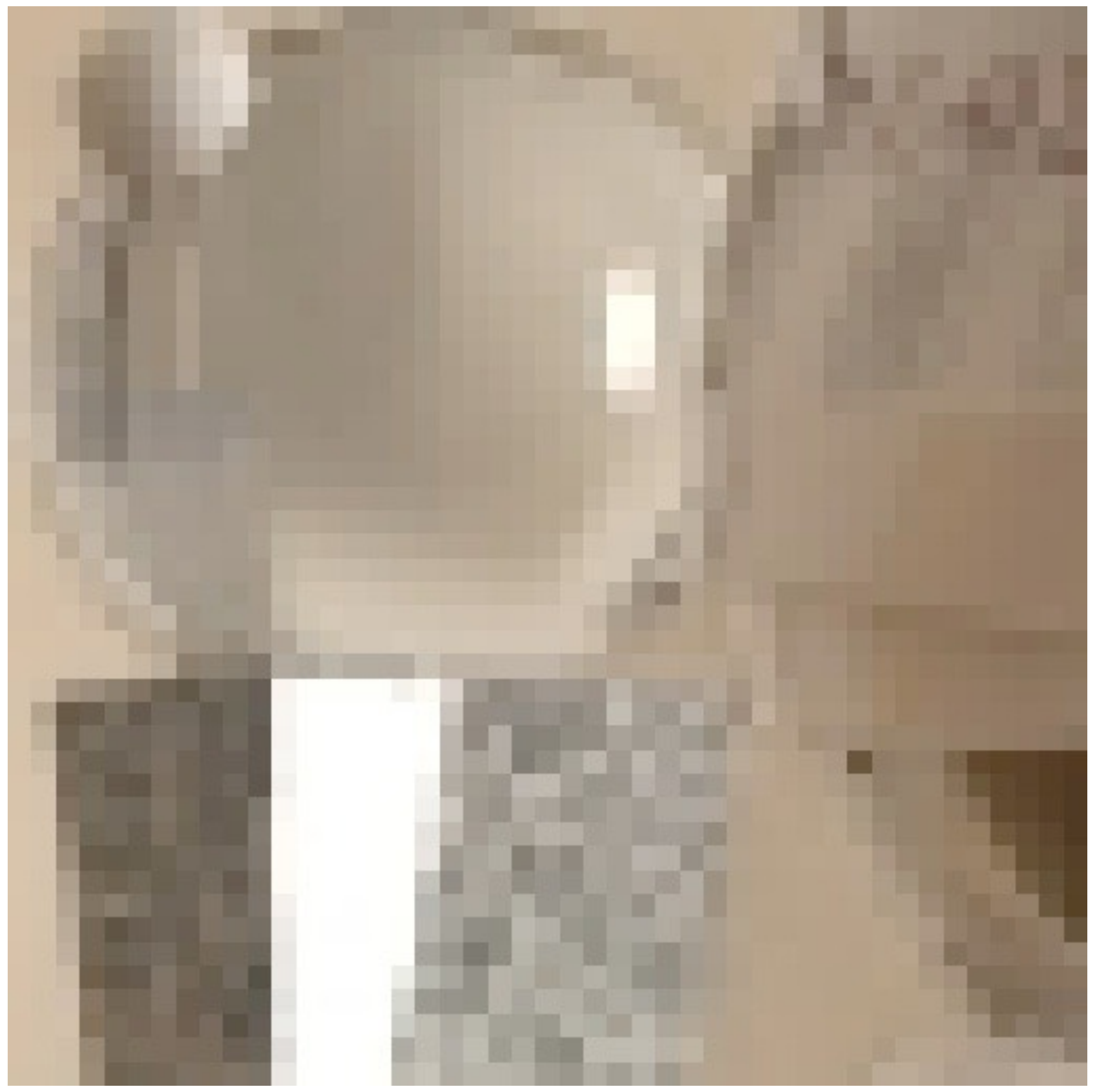

Final color upsampling

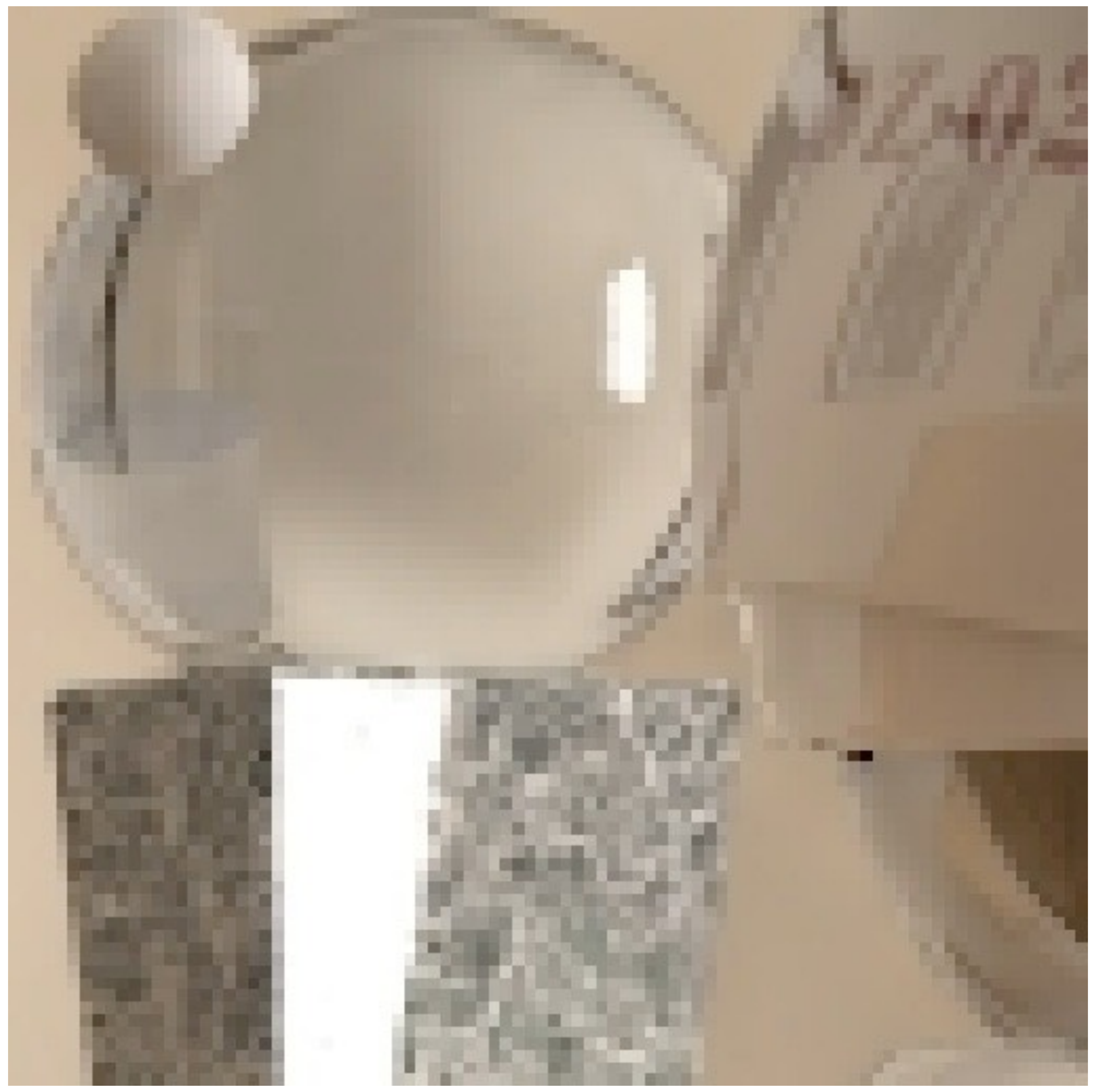

Ground truth

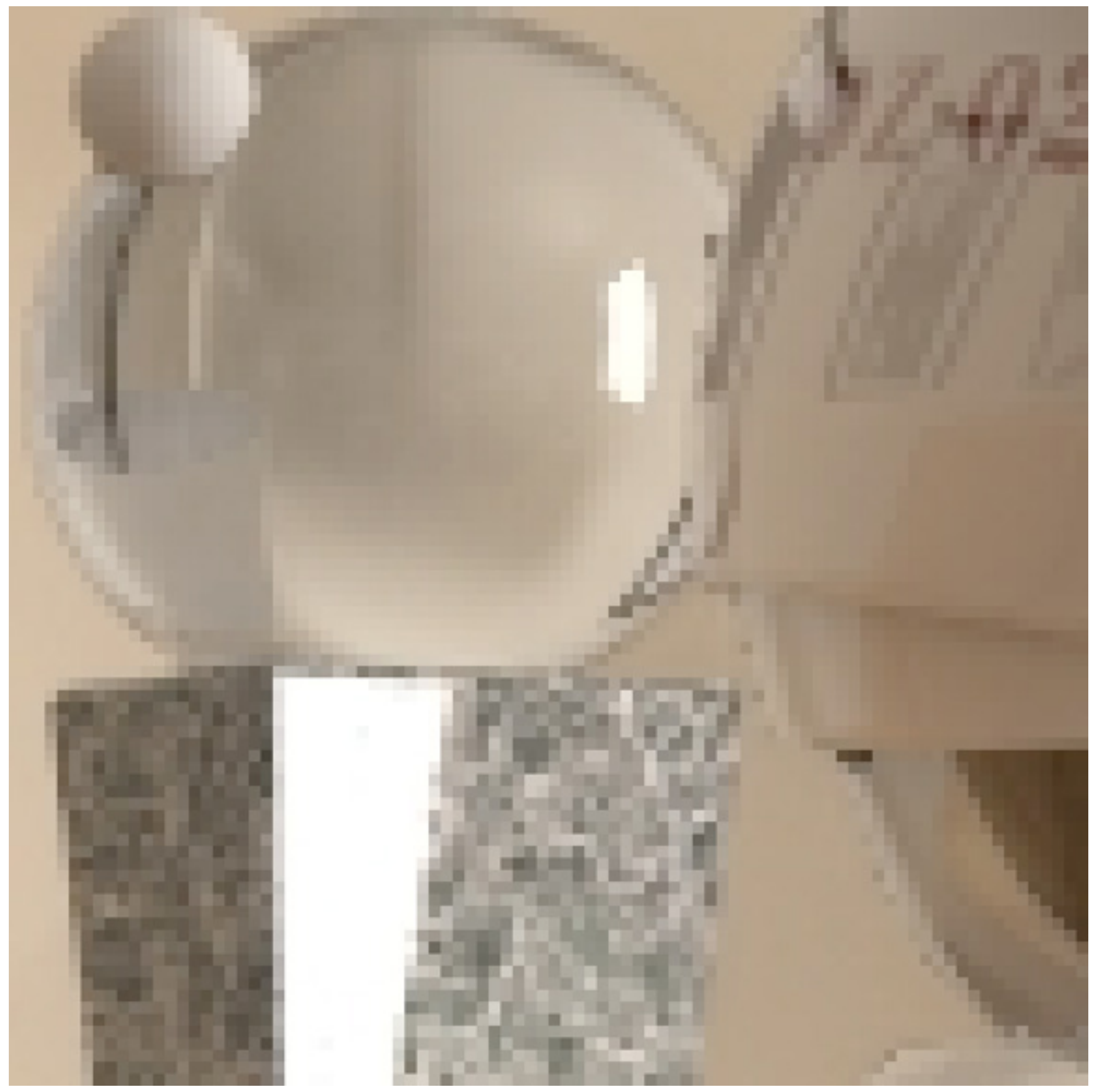
Similar to the denoising results, a standard upsampling of the final pixel colors guided by high-resolution geometry features successfully improves the resolution, as shown in the middle.

However, it misses almost all details of the objects seen through the glass sphere when compared to the ground truth on the right

Spatial Upsampling


Low-res input

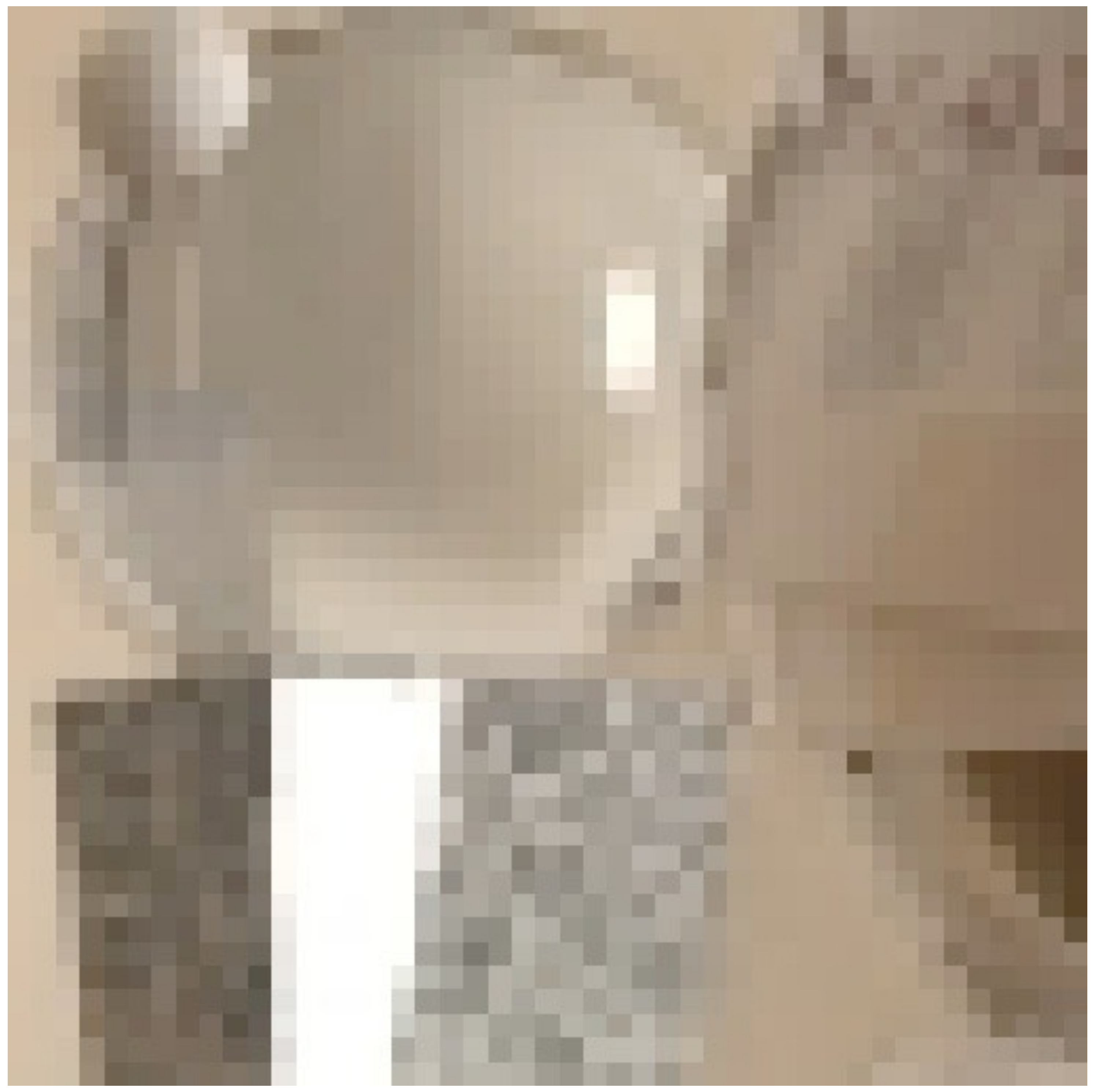

Per-component upsampling

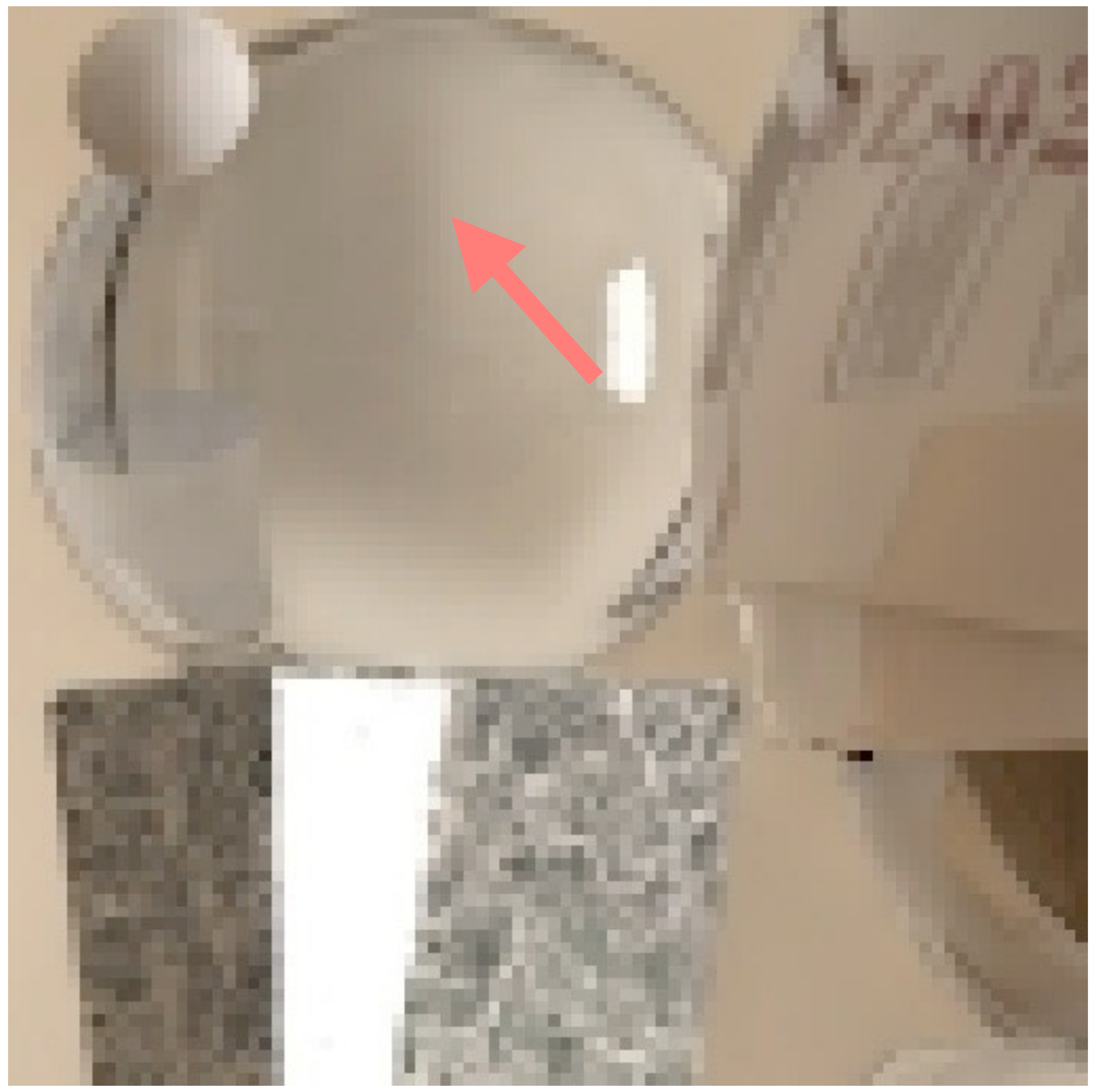

Ground truth

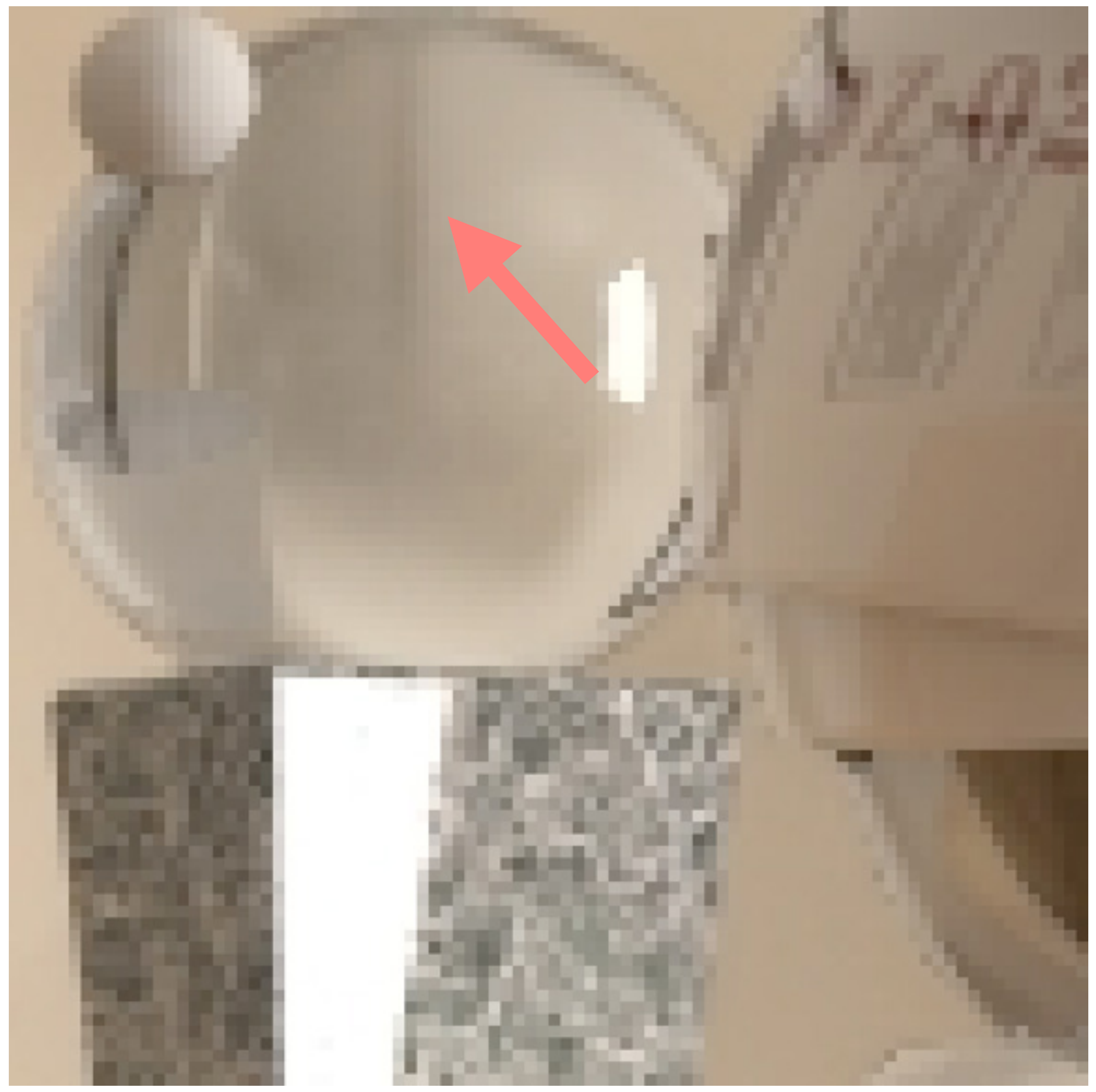
Our per-component upsampling using per-component high-resolution features as a guide, preserves these details.
<switch back and forth>

<click>
Compared to the ground truth, we only miss the weak shadow marked by the arrow.
And this is because shadows are not captured by our per-component features.

Spatial Upsampling


Low-res input


Per-component upsampling


Ground truth

Our per-component upsampling using per-component high-resolution features as a guide, preserves these details.
<switch back and forth>

<click>
Compared to the ground truth, we only miss the weak shadow marked by the arrow.
And this is because shadows are not captured by our per-component features.

Performance

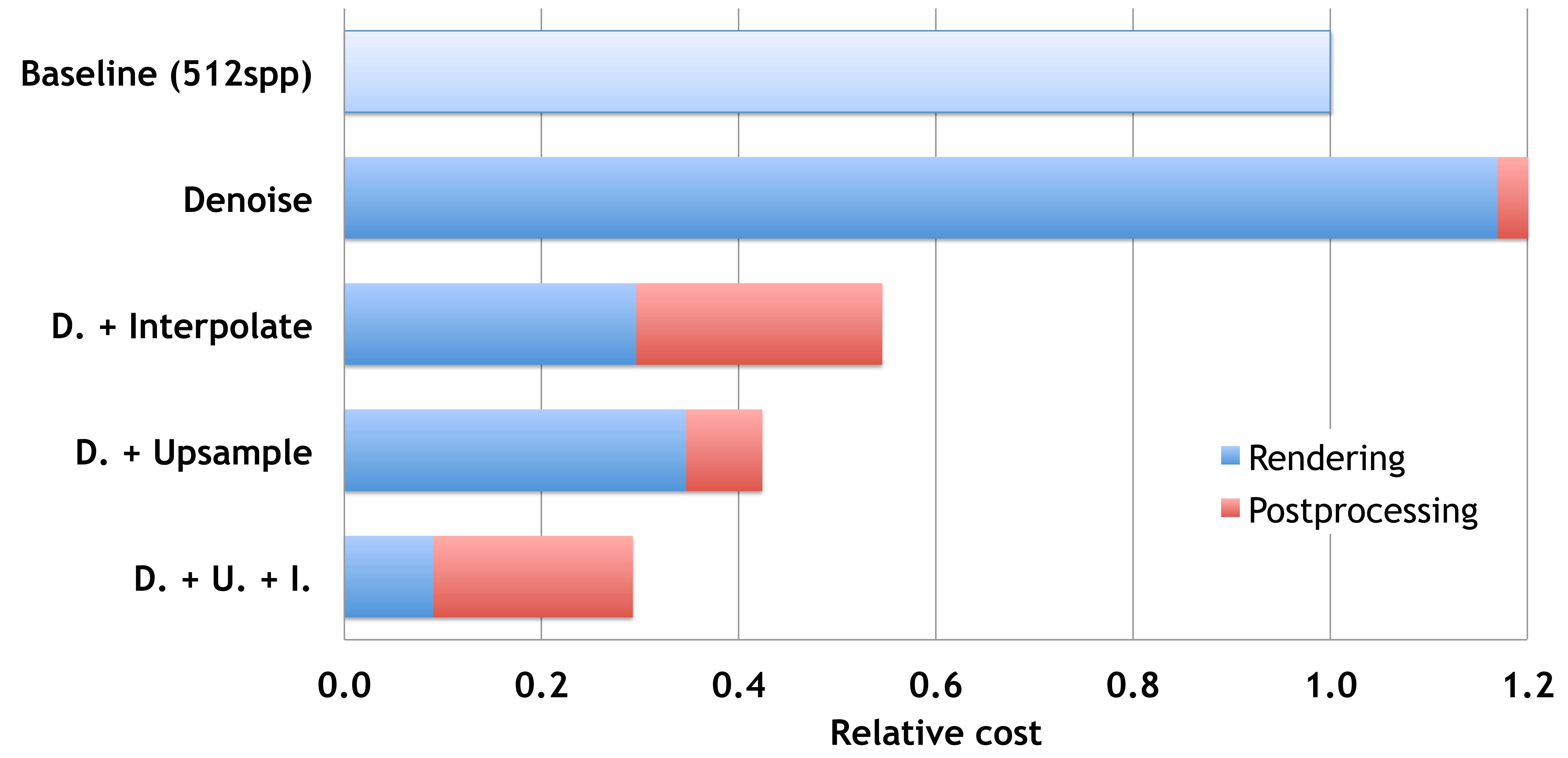
Noisy baseline at 512spp

$$\textbf{relative cost} = \frac{\textbf{overhead w.r.t. baseline}}{\textbf{resolution gain}}$$

To measure performance, we compare computation times against a noisy baseline render with 512 spp.

We then compute the relative cost,
that is the computational overhead w.r.t. the baseline divided by the possible resolution gain due to upsampling and/or frame interpolation.

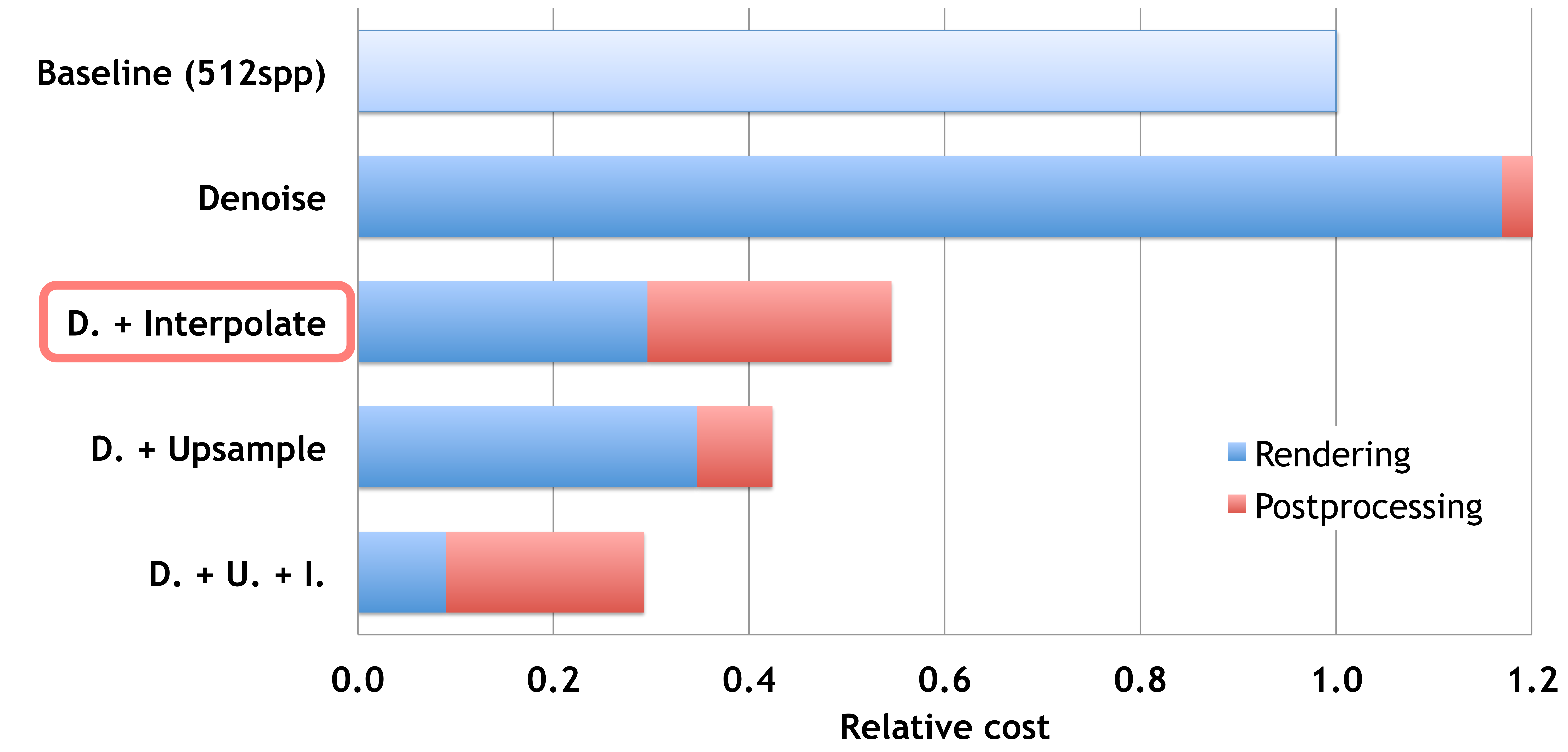
Performance



Monday, June 29, 15

Here we plot the relative costs, and distinguish between the costs for rendering the input data in blue and potential post-processing steps in red.
<click>
As one can see, already combining denoising and interpolation reduces the relative cost to about 0.5
<click>
and when combining all three steps, the relative cost goes down to about 0.3.

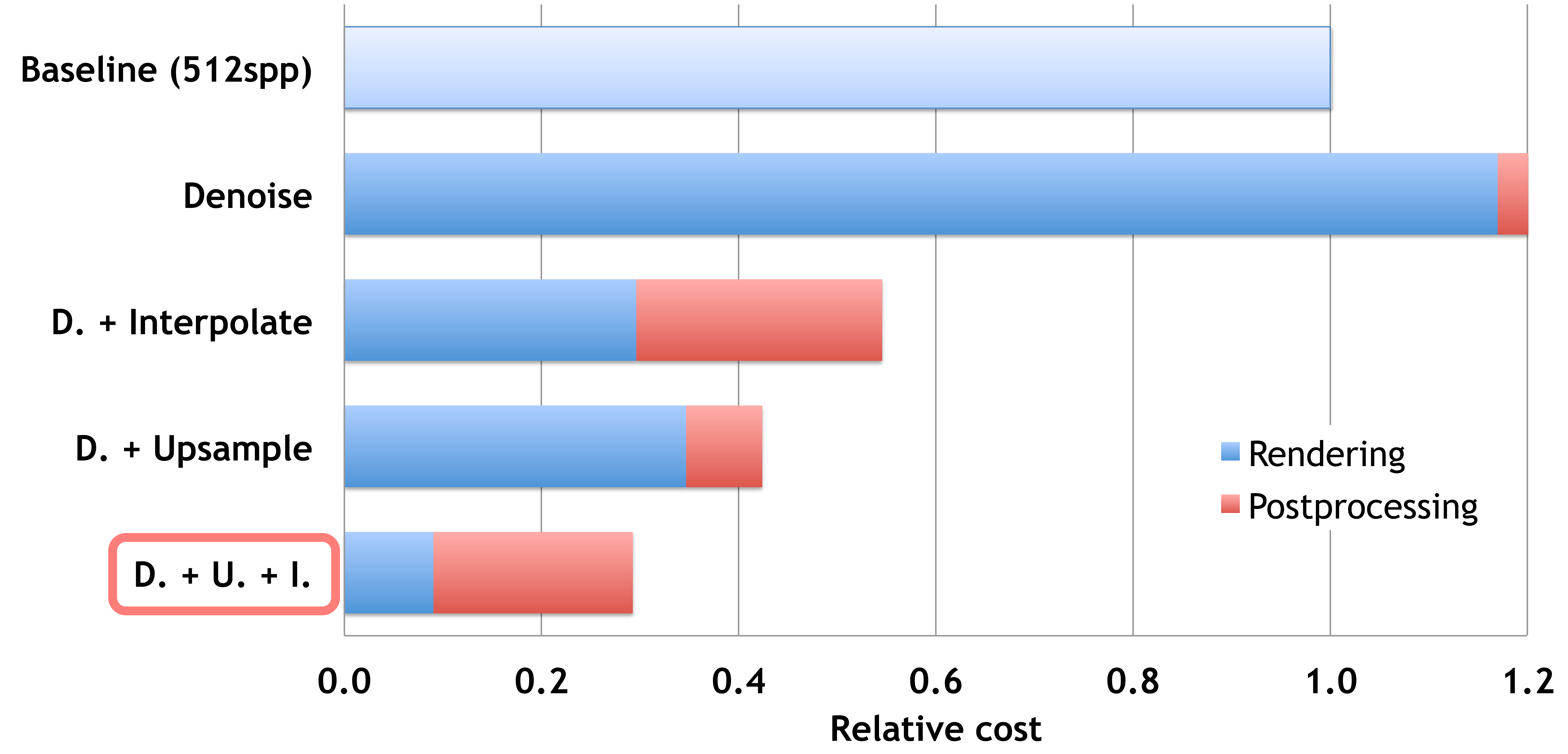
Performance



Monday, June 29, 15

Here we plot the relative costs, and distinguish between the costs for rendering the input data in blue and potential post-processing steps in red.
<click>
As one can see, already combining denoising and interpolation reduces the relative cost to about 0.5
<click>
and when combining all three steps, the relative cost goes down to about 0.3.

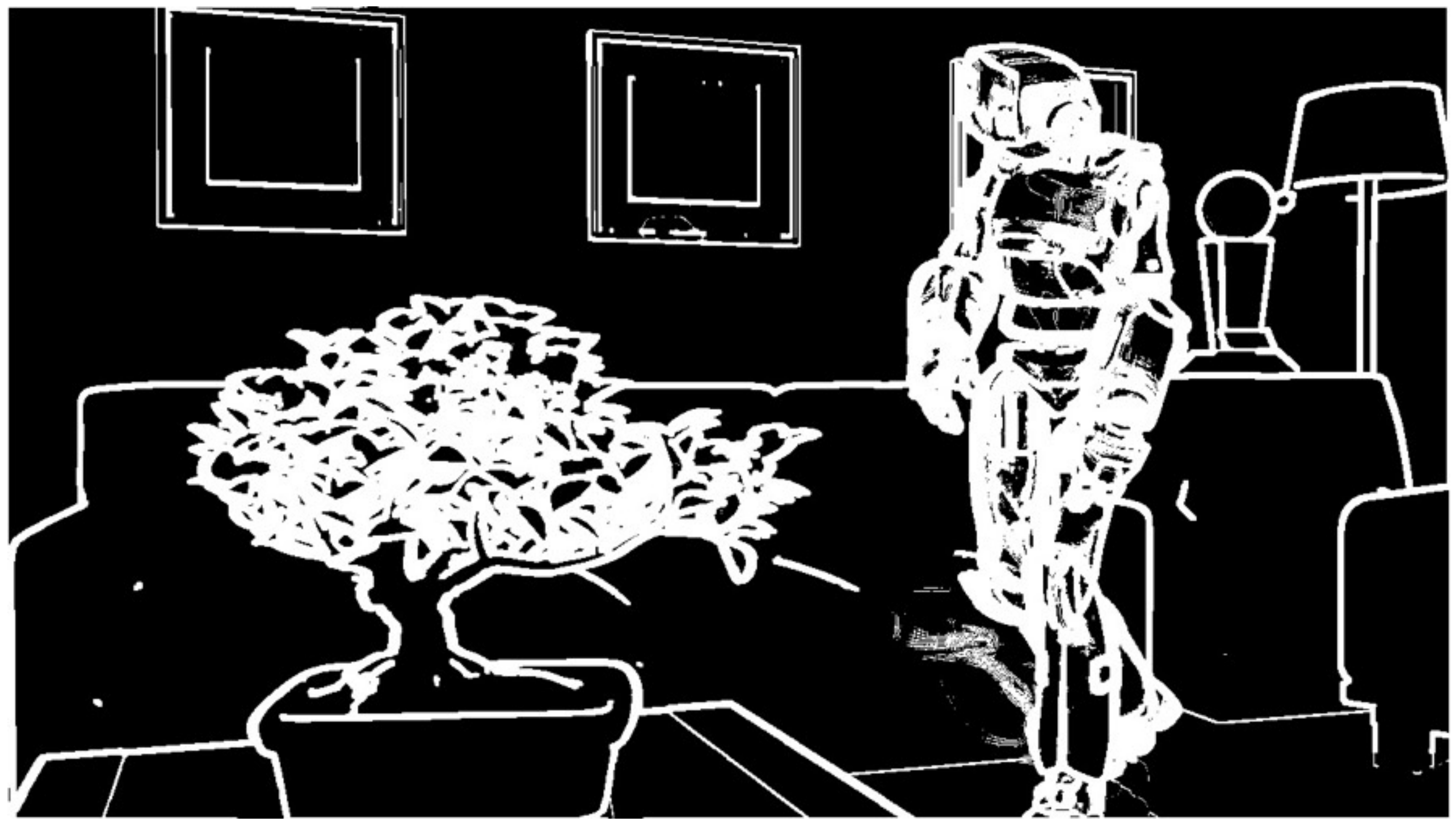
Performance



Here we plot the relative costs, and distinguish between the costs for rendering the input data in blue and potential post-processing steps in red.
<click>
As one can see, already combining denoising and interpolation reduces the relative cost to about 0.5
<click>
and when combining all three steps, the relative cost goes down to about 0.3.

Limitations

Sliver re-rendering for interpolation



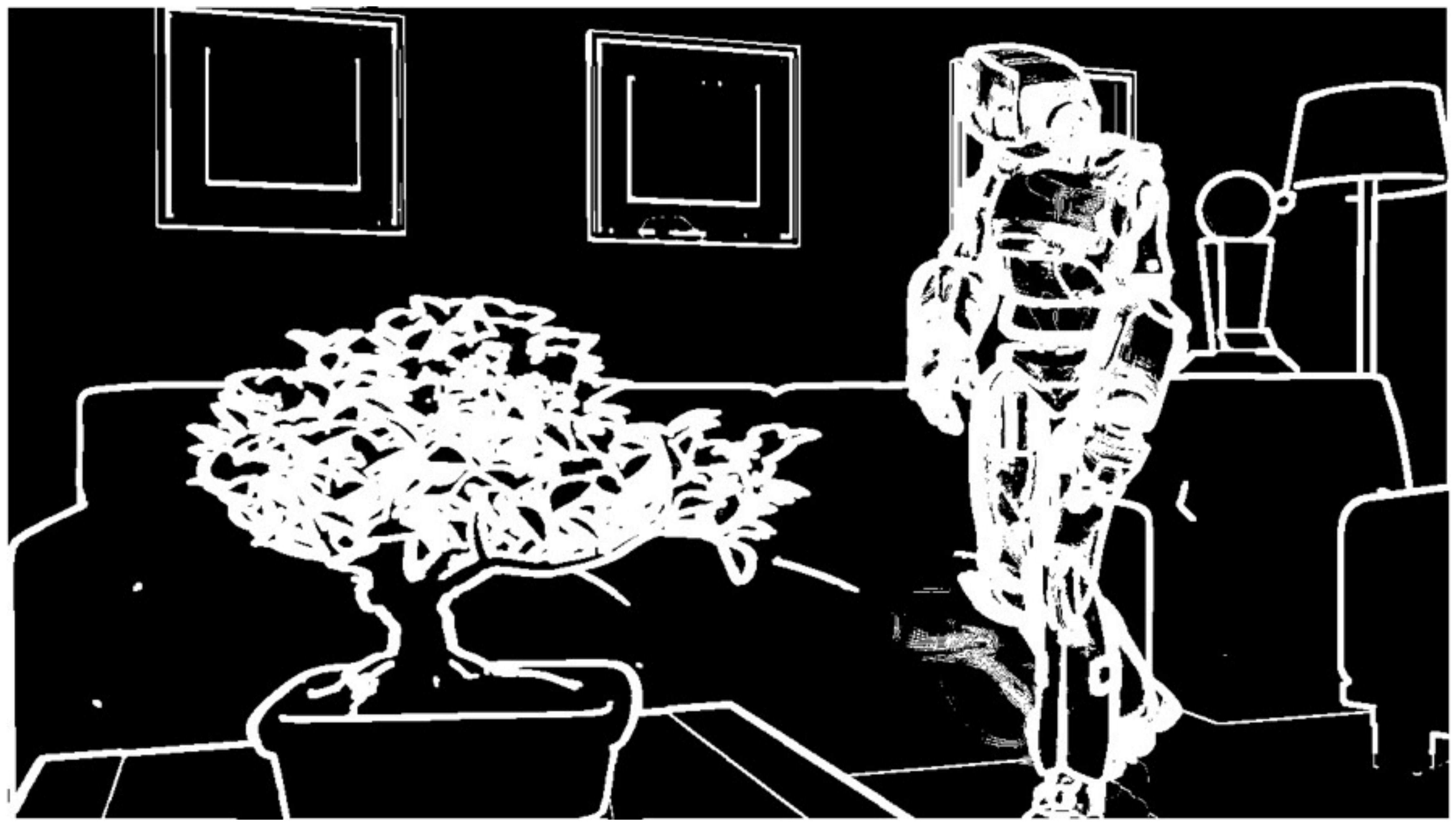
Our method also has some limitations.

One problem is that some pixels cannot be handled for frame interpolation and need to be rendered in a second, sparse re-rendering pass.

<click>
For the robot scene, the average re-rendering rate was 20%

Limitations

Sliver re-rendering for interpolation



Re-rendering rate:
20% on average

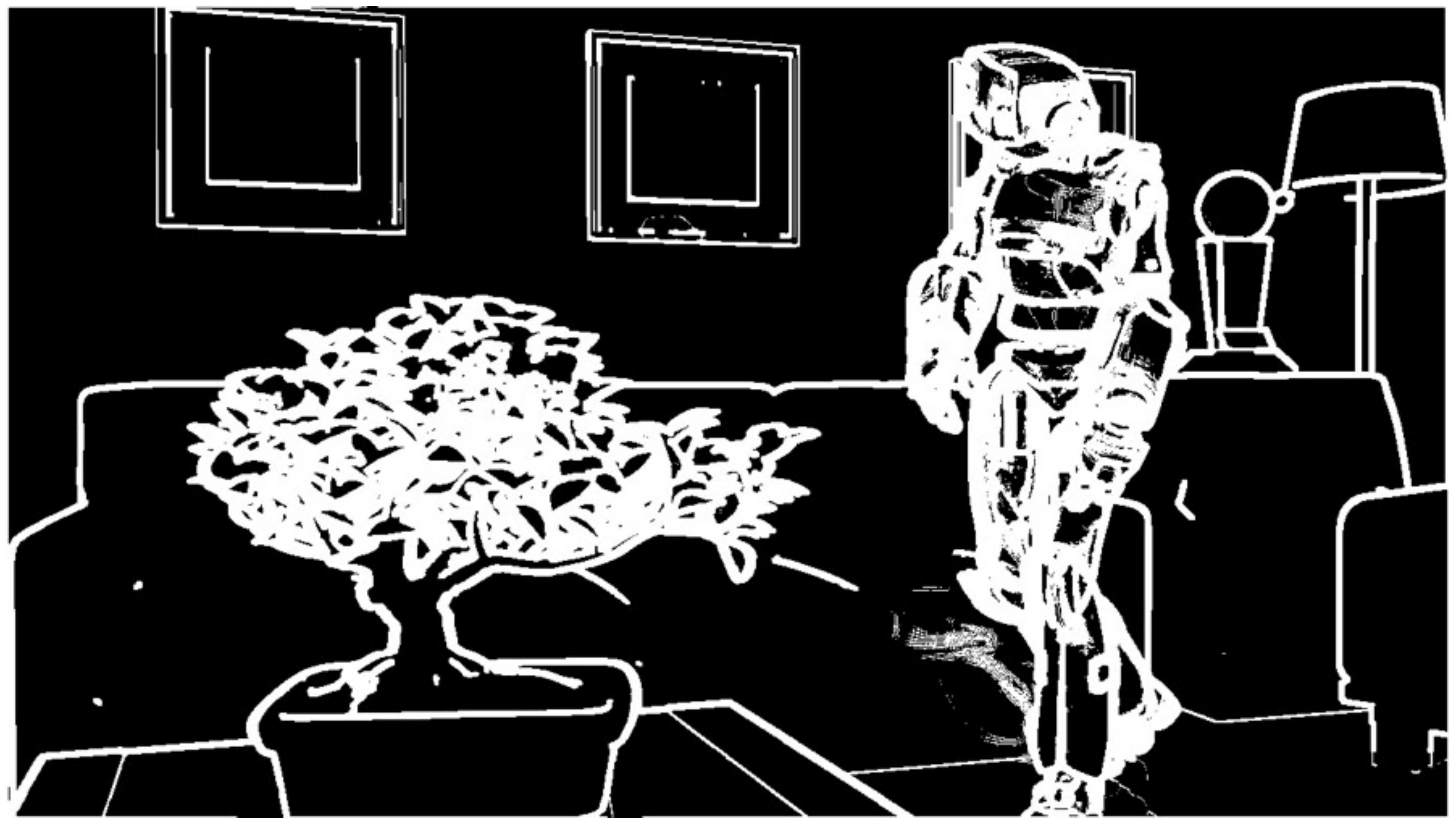
Our method also has some limitations.

One problem is that some pixels cannot be handled for frame interpolation and need to be rendered in a second, sparse re-rendering pass.

<click>
For the robot scene, the average re-rendering rate was 20%

Limitations

Sliver re-rendering for interpolation



Re-rendering rate:
20% on average

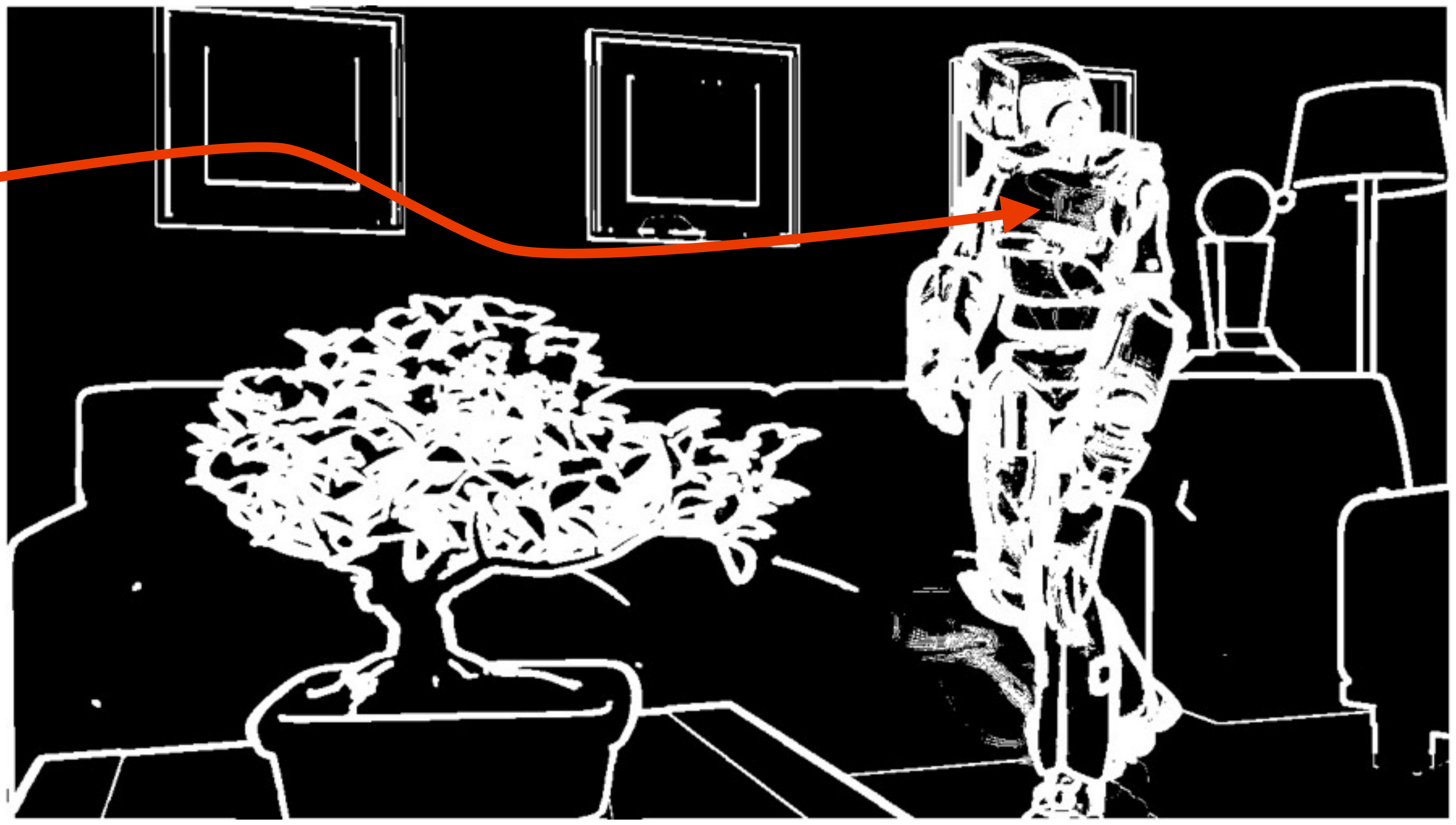
There are several reasons why pixels need to be rerendered.

As mentioned before, one reason are
<click>
invalid specular paths where no specular motion vectors could be found.
Other reasons are
<click> occlusions
<click> as well as silhouette pixels,

Limitations

Sliver re-rendering for interpolation

invalid specular paths



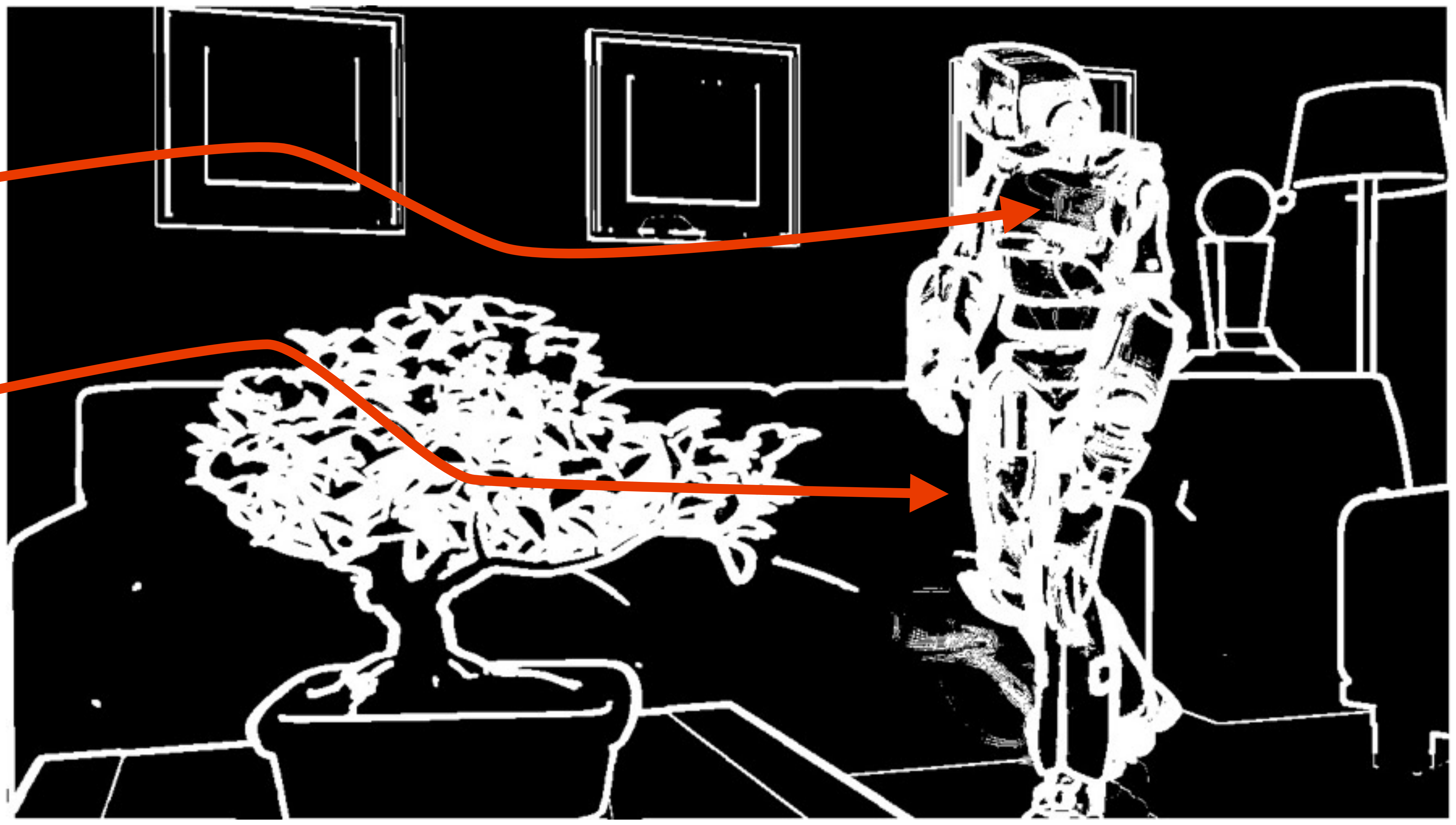
Re-rendering rate: 20% on average

There are several reasons why pixels need to be rerendered.

As mentioned before, one reason are
<click>
invalid specular paths where no specular motion vectors could be found.
Other reasons are
<click> occlusions
<click> as well as silhouette pixels,
as detailed in our paper

Limitations

Sliver re-rendering for interpolation



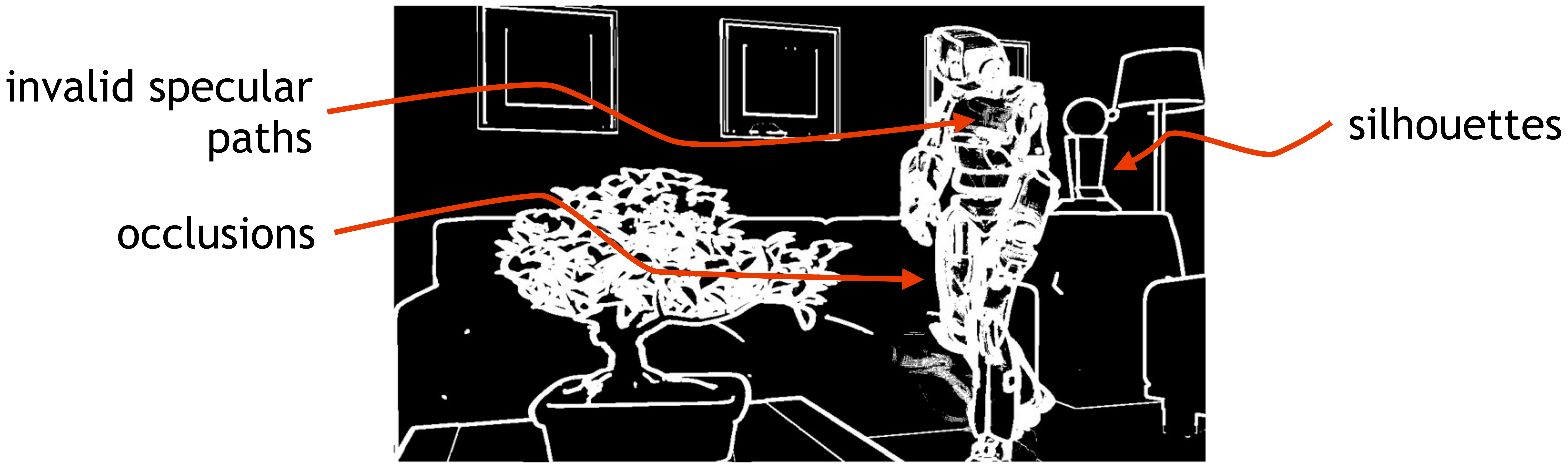
Re-rendering rate: 20% on average

There are several reasons why pixels need to be rerendered.

As mentioned before, one reason are
<click>
invalid specular paths where no specular motion vectors could be found.
Other reasons are
<click> occlusions
<click> as well as silhouette pixels,
as detailed in our paper

Limitations

Sliver re-rendering for interpolation



Re-rendering rate:
20% on average

There are several reasons why pixels need to be rerendered.

As mentioned before, one reason are
<click>
invalid specular paths where no specular motion vectors could be found.
Other reasons are
<click> occlusions
<click> as well as silhouette pixels,
as detailed in our paper

Limitations

Sliver re-rendering for interpolation

Interpolation with Depth-of-Field and Motion Blur

- Problem: motion vectors are ill-defined
- Solution: simulate in post-process

Lastly, renderings with Depth-of-field or motion blur cannot be handled for frame interpolation
as in these cases, a single motion vector cannot describe the motion of blurred pixels capturing different scene objects with potentially different motions.

As a remedy, we can render images without these effects and simulate them as a post-process.

Post-Process Motion Blur

which we exemplify here by simulating motion blur.
<click>
To simulate motion blur, we simply integrate the pixel colors of each component along its motion vectors.

Post-Process Motion Blur

which we exemplify here by simulating motion blur.
<click>
To simulate motion blur, we simply integrate the pixel colors of each component along its motion vectors.

Conclusion

Decomposition: disambiguate input in image-space

In conclusion, we have presented a general decomposition framework that allows us to disambiguate secondary lighting effects, such as reflections and moving shadows, in image space.

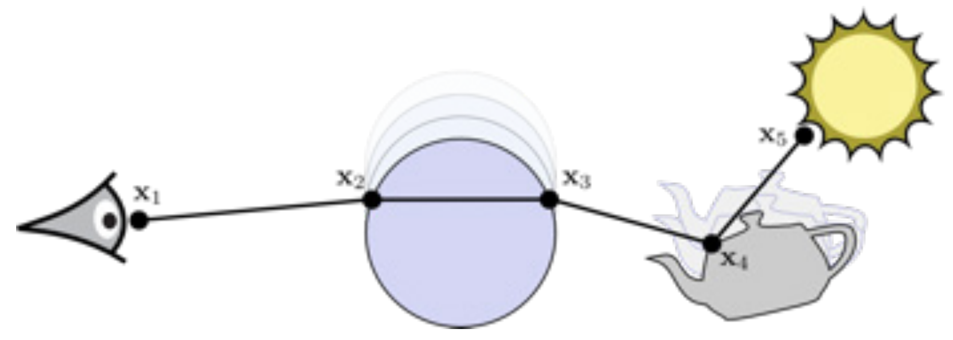
We showed that image-space post-processing methods such as frame interpolation, denoising, and spatial upsampling can largely benefit from such a decomposition, especially when leveraging per-component features and motion vectors.

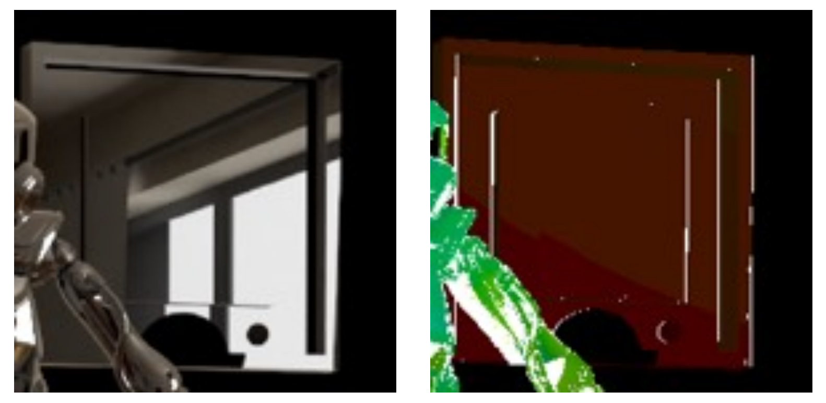
One big advantage of our approach is that it does not require to change the underlying post-processing algorithms, we just need to apply them for each component separately.

Conclusion

Decomposition: disambiguate input in image-space

Temporal manifold exploration for specular motion



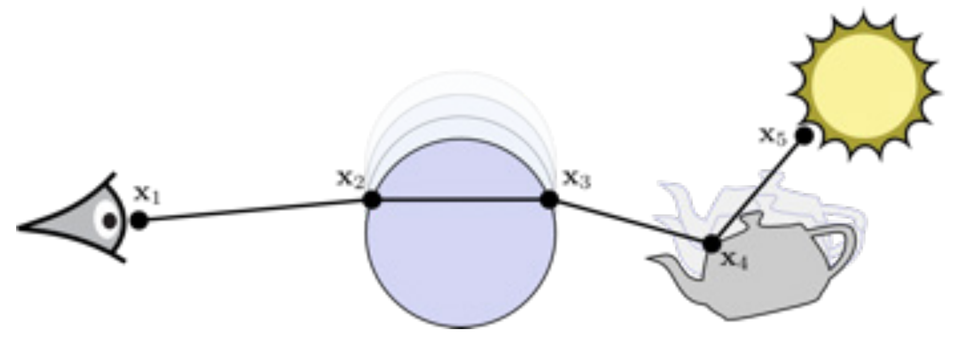


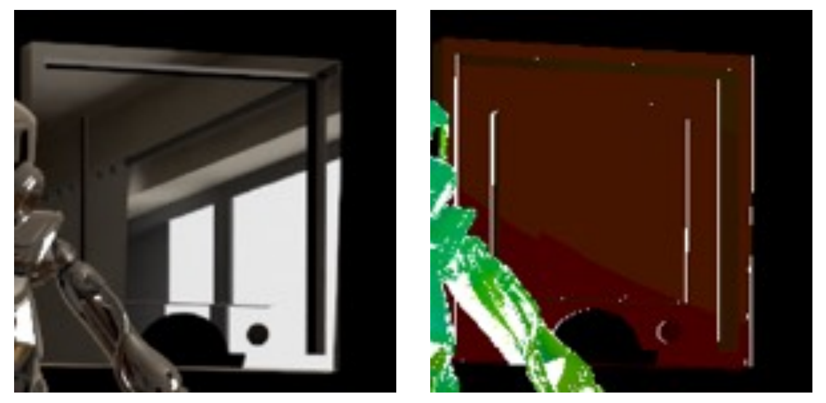
To compute highly accurate motion vectors for arbitrary specular transport,
we extended manifold exploration to the time dimension.

Conclusion

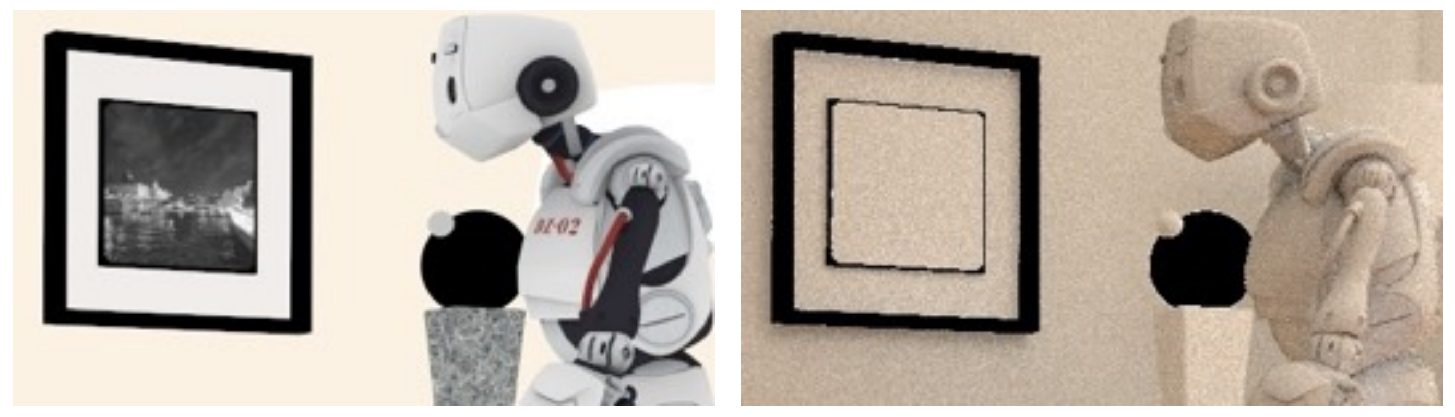
Decomposition: disambiguate input in image-space

Temporal manifold exploration for specular motion





Effective irradiance factorization & motion estimation



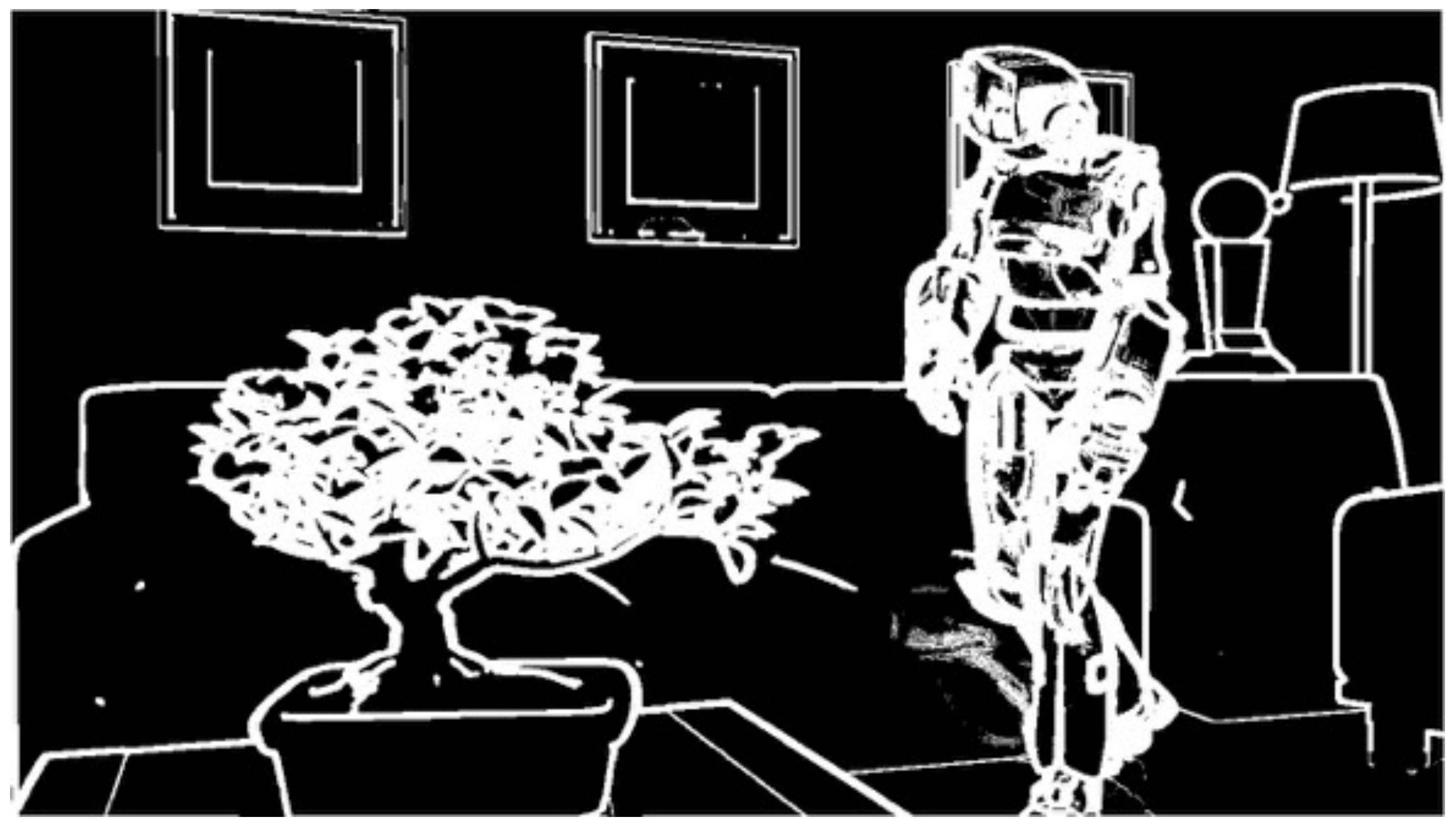
Monday, June 29, 15

Finally, we presented an effective irradiance factorization to separate texture from lighting in the presence of non-diffuse surfaces

and showed how optical flow can be used to estimate corresponding irradiance motion vectors.

Future Work

Reduce re-rendering for interpolation

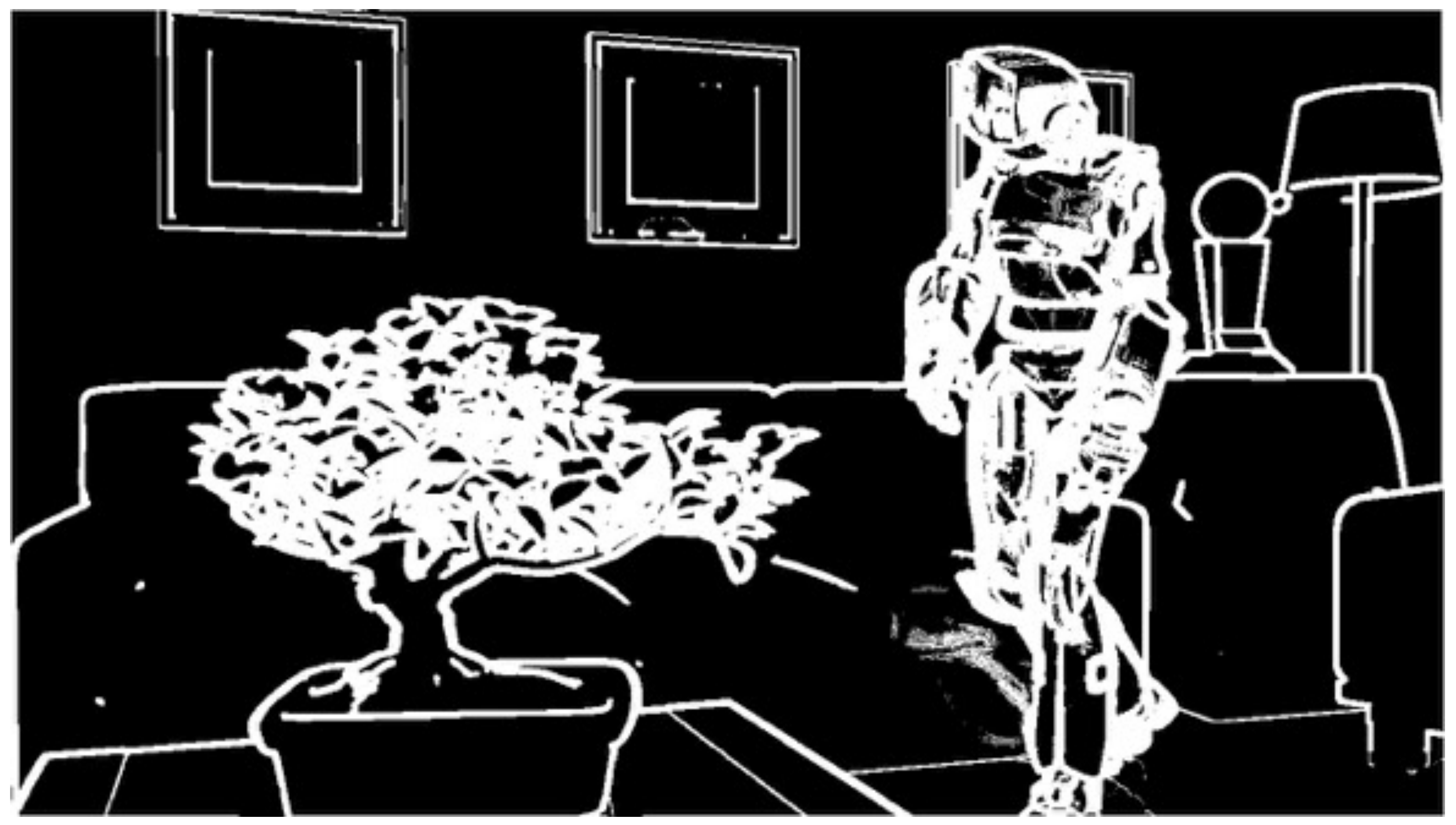


In future work, we aim to investigate ways to reduce the amount of re-rendering needed for frame interpolation

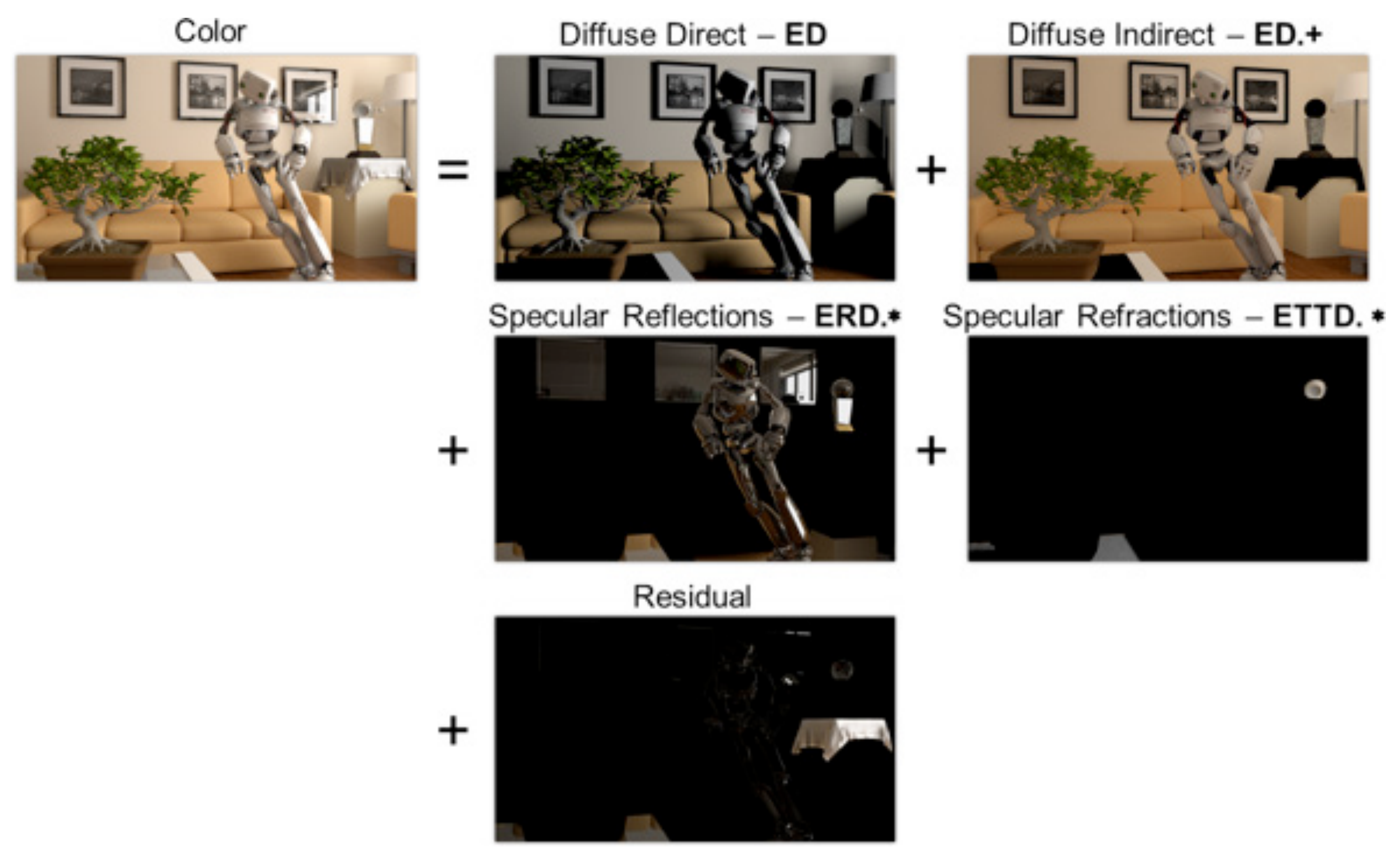
<click>
as well as developing an automatic method for determining the optimal granularity of our decomposition.

Future Work

Reduce re-rendering for interpolation



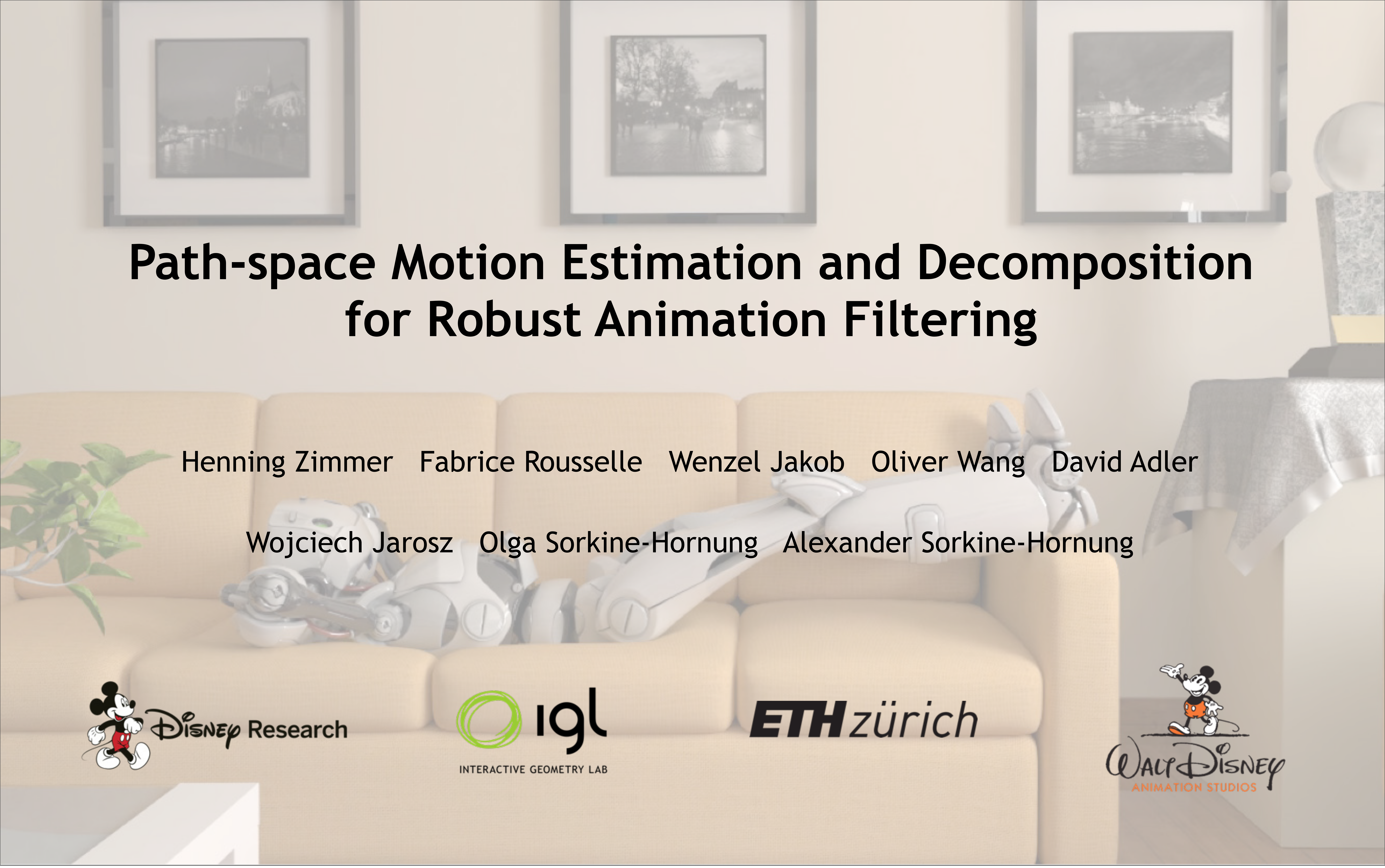
Automatic decomposition granularity



Monday, June 29, 15

In future work, we aim to investigate ways to reduce the amount of re-rendering needed for frame interpolation

<click>
as well as developing an automatic method for determining the optimal granularity of our decomposition.



Path-space Motion Estimation and Decomposition for Robust Animation Filtering

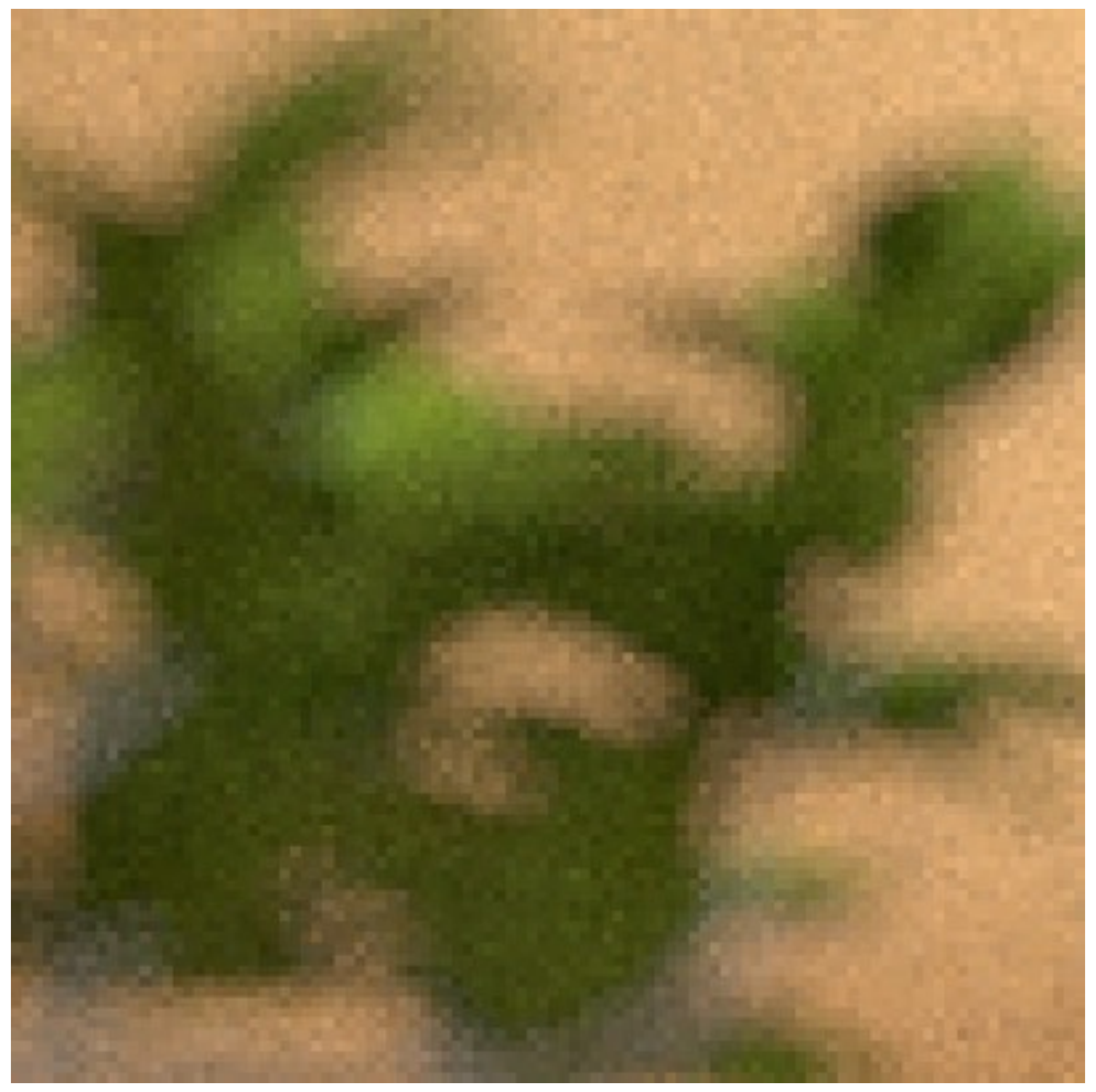
Henning Zimmer   Fabrice Rousselle   Wenzel Jakob   Oliver Wang   David Adler

Wojciech Jarosz   Olga Sorkine-Hornung   Alexander Sorkine-Hornung

Disney Research   igl INTERACTIVE GEOMETRY LAB   ETH zürich   Walt Disney ANIMATION STUDIOS

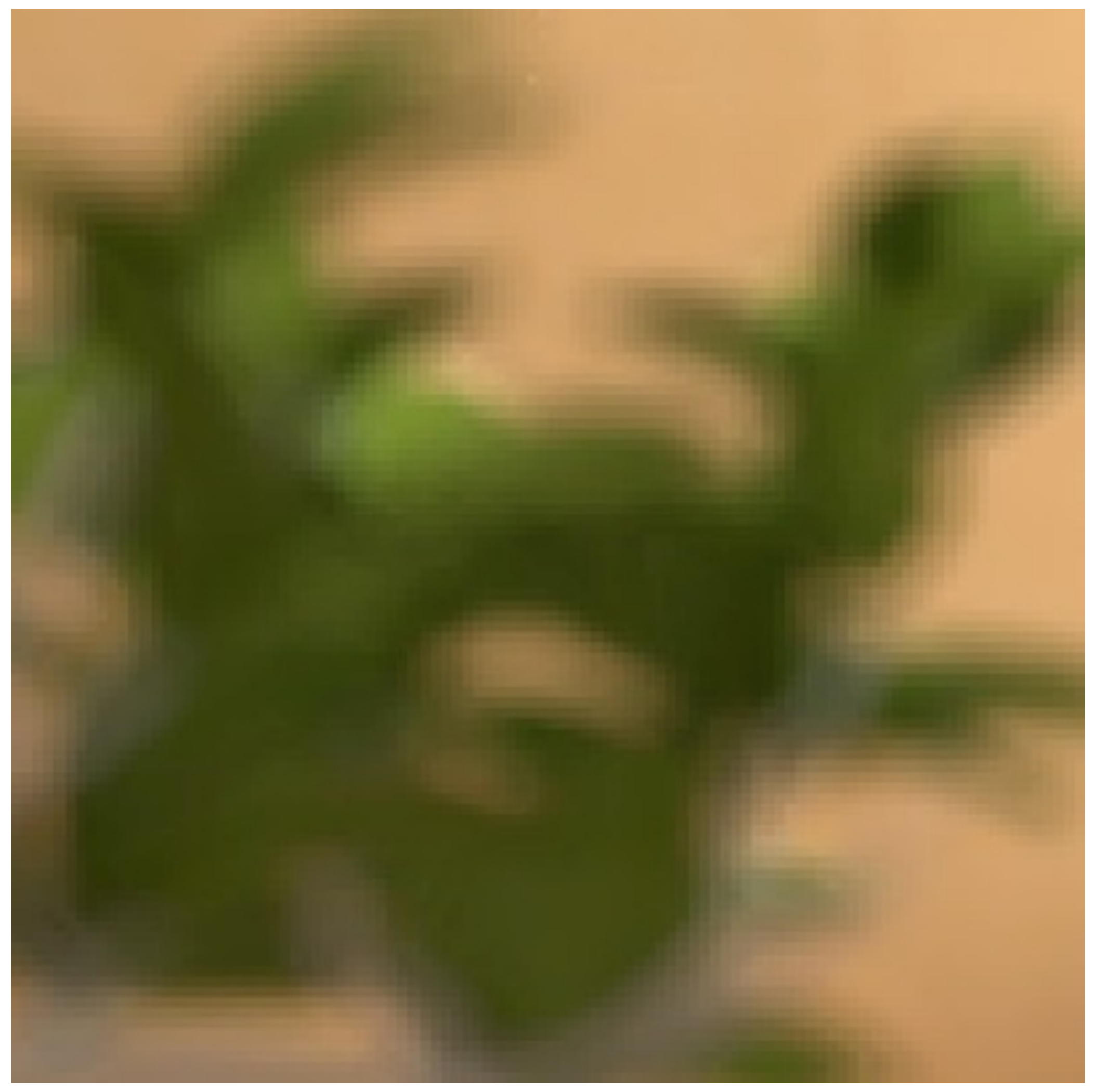
Irradiance Factorization and Depth-of-Field

Input (512 spp)



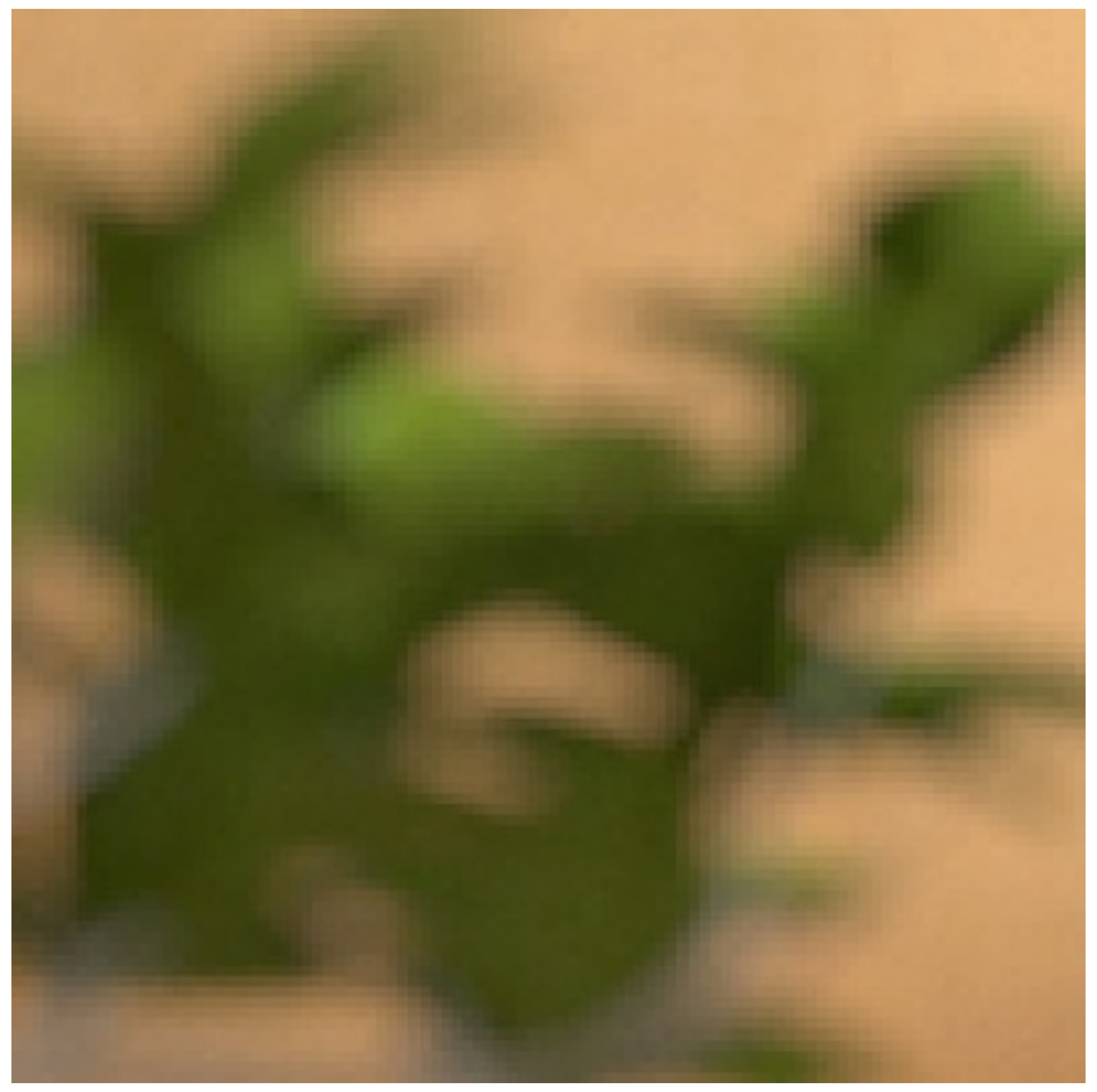
rel. MSE: $5.6 \cdot 10^{-3}$

Std. irradiance



$9.9 \cdot 10^{-3}$

Ground truth



To illustrate the robustness of our effective irradiance factorization, we added depth-of-field to our robot scene and then ran our denoising filter. The result shown here is a crop of the bonsai.

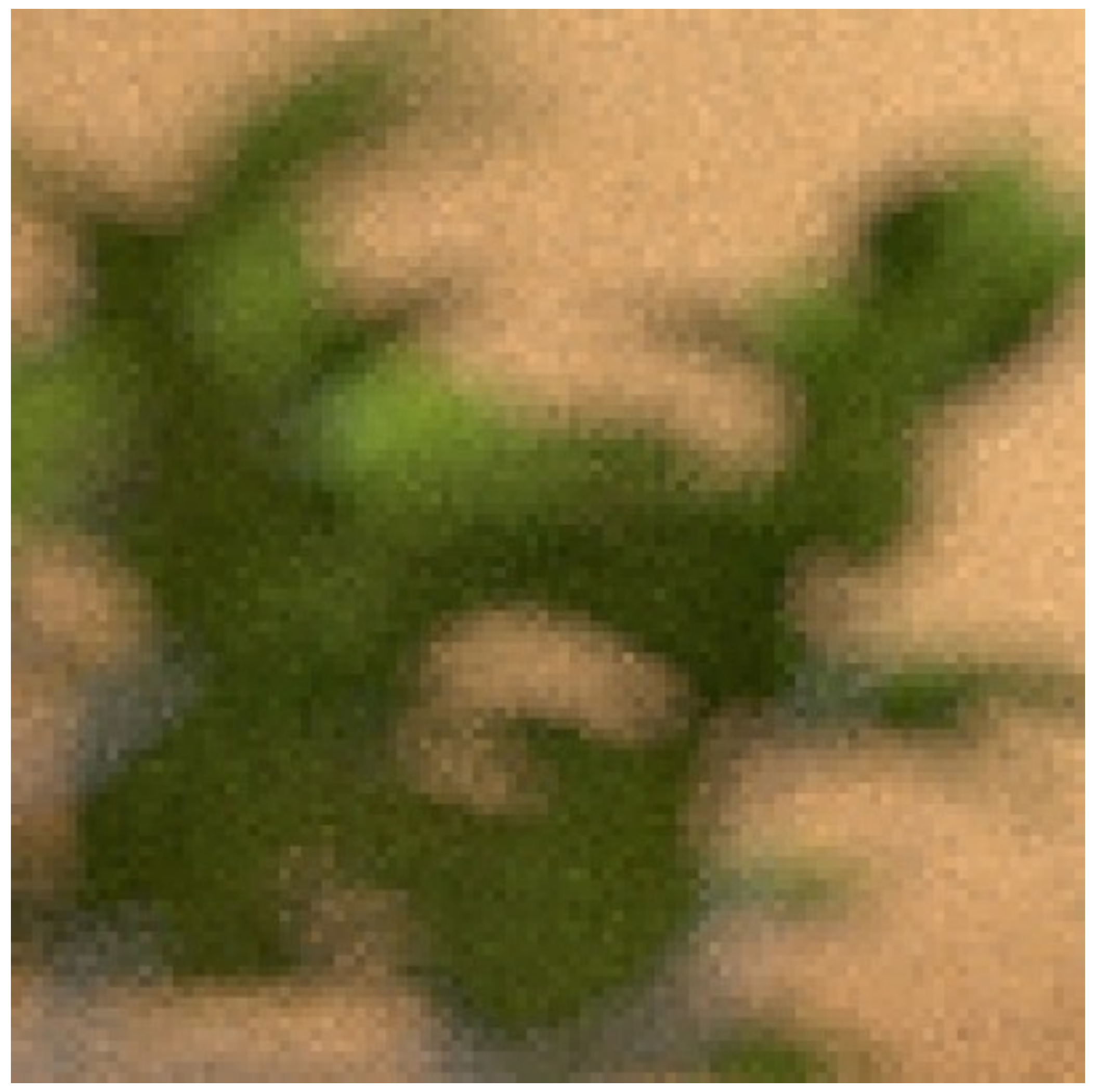
Even though both the bonsai and the sofa in the background are Lambertian surfaces, the depth-of-field effect invalidates the assumption of a linear shading model that standard irradiance factorization relies upon.

While the standard irradiance factorization correctly handles uniform regions, regions with mixed reflectances, such as the one pointed by the arrow, suffer from significant bias. This bias is clearly visible in the bottom row, where we directly visualize the relative squared error, and explains why the filtered output has a higher MSE than the noisy input.

In contrast, our effective irradiance factorization robustly handles regions with mixed reflectances and yields a much better reconstruction overall.

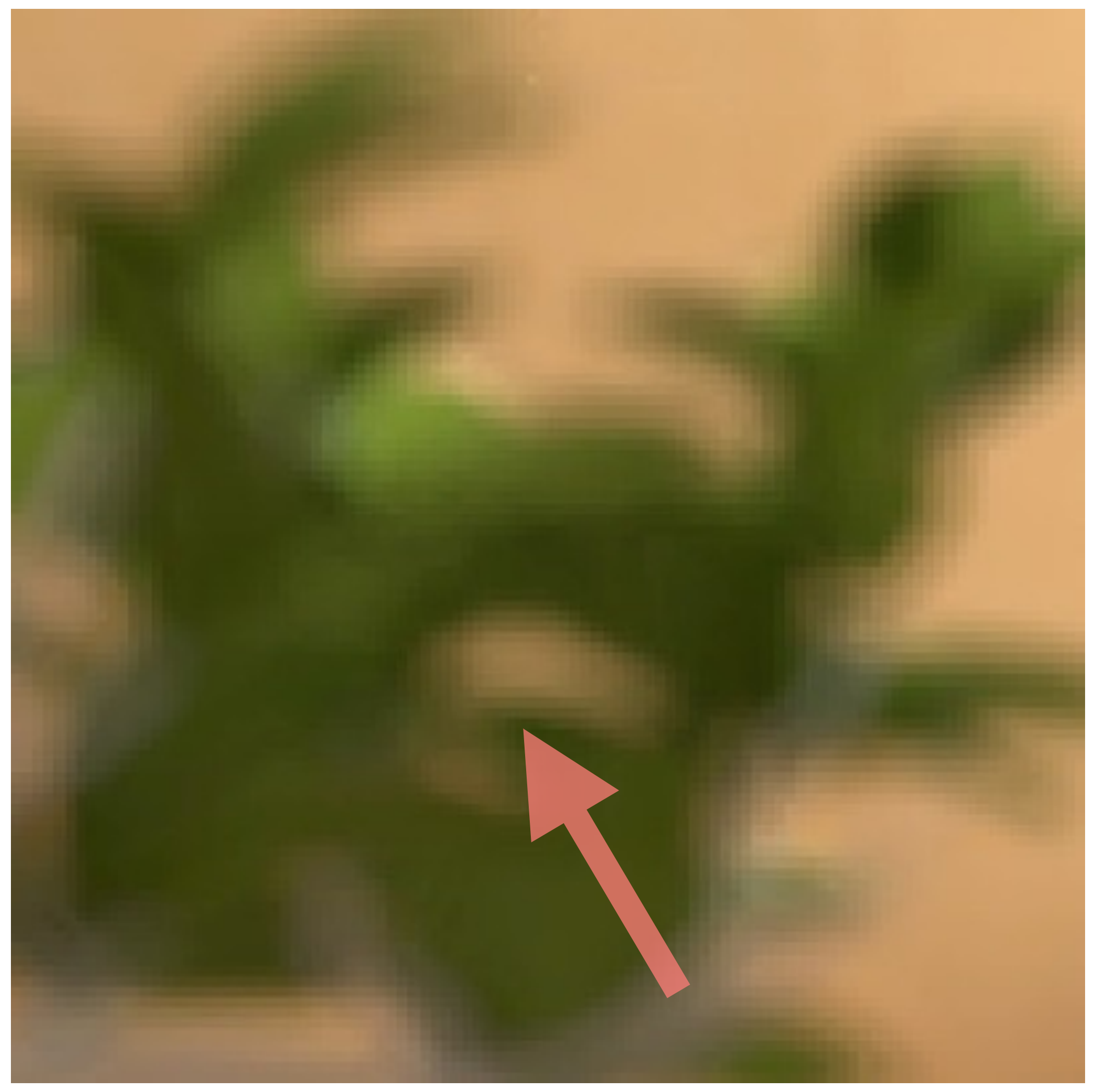
Irradiance Factorization and Depth-of-Field

Input (512 spp)



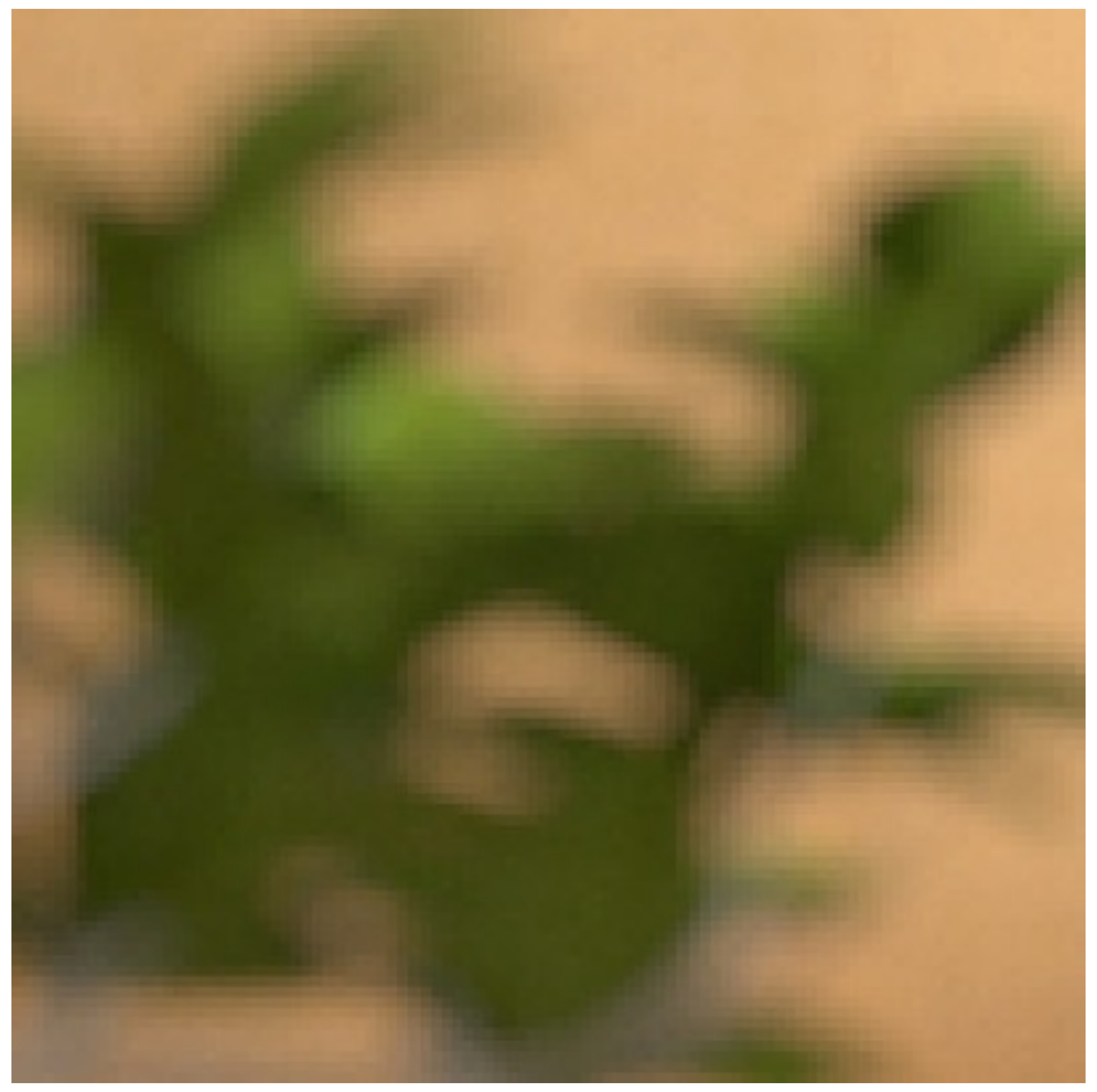
rel. MSE: $5.6 \cdot 10^{-3}$

Std. irradiance



$9.9 \cdot 10^{-3}$

Ground truth



To illustrate the robustness of our effective irradiance factorization, we added depth-of-field to our robot scene and then ran our denoising filter. The result shown here is a crop of the bonsai.

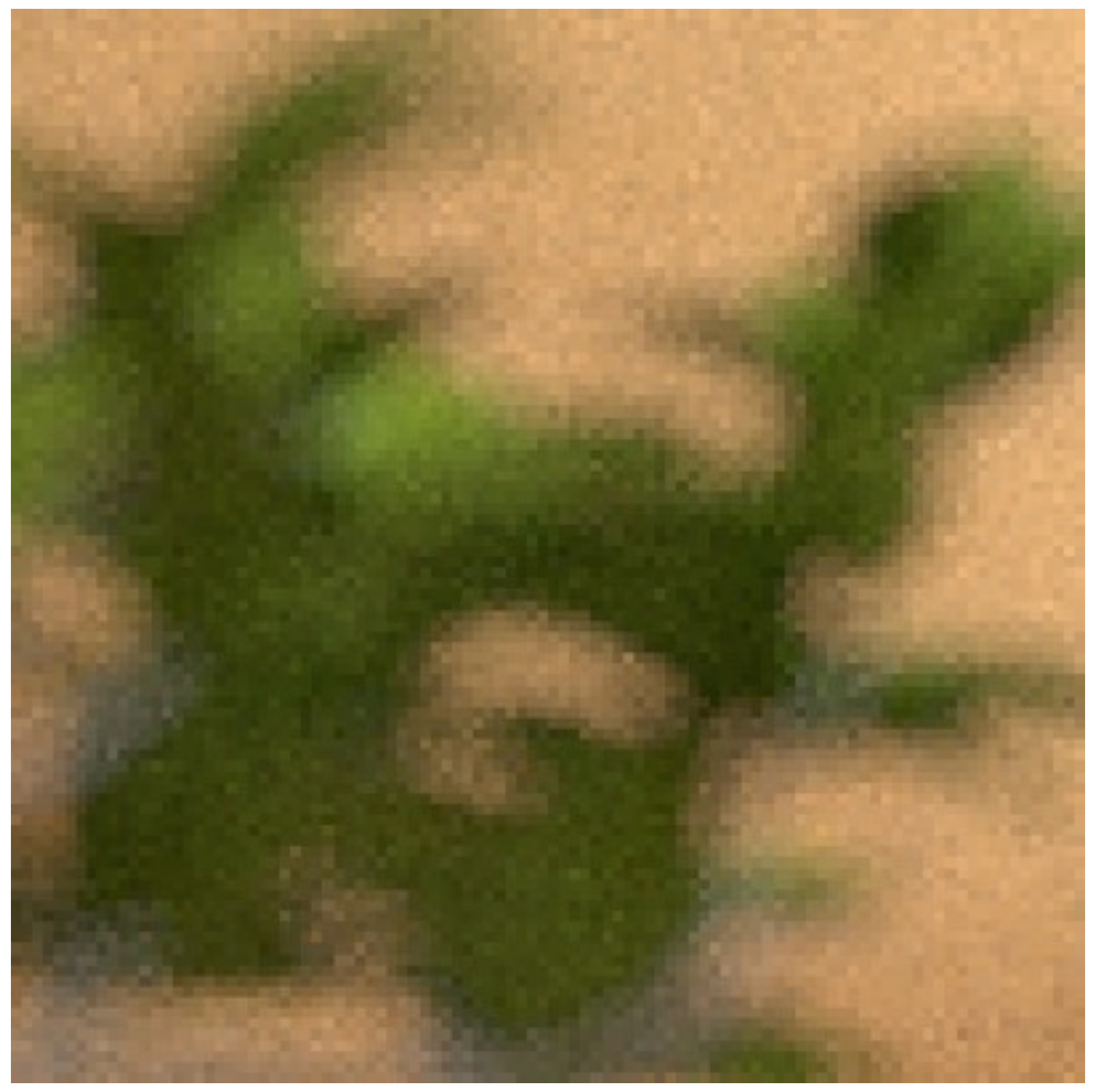
Even though both the bonsai and the sofa in the background are Lambertian surfaces, the depth-of-field effect invalidates the assumption of a linear shading model that standard irradiance factorization relies upon.

While the standard irradiance factorization correctly handles uniform regions, regions with mixed reflectances, such as the one pointed by the arrow, suffer from significant bias. This bias is clearly visible in the bottom row, where we directly visualize the relative squared error, and explains why the filtered output has a higher MSE than the noisy input.

In contrast, our effective irradiance factorization robustly handles regions with mixed reflectances and yields a much better reconstruction overall.

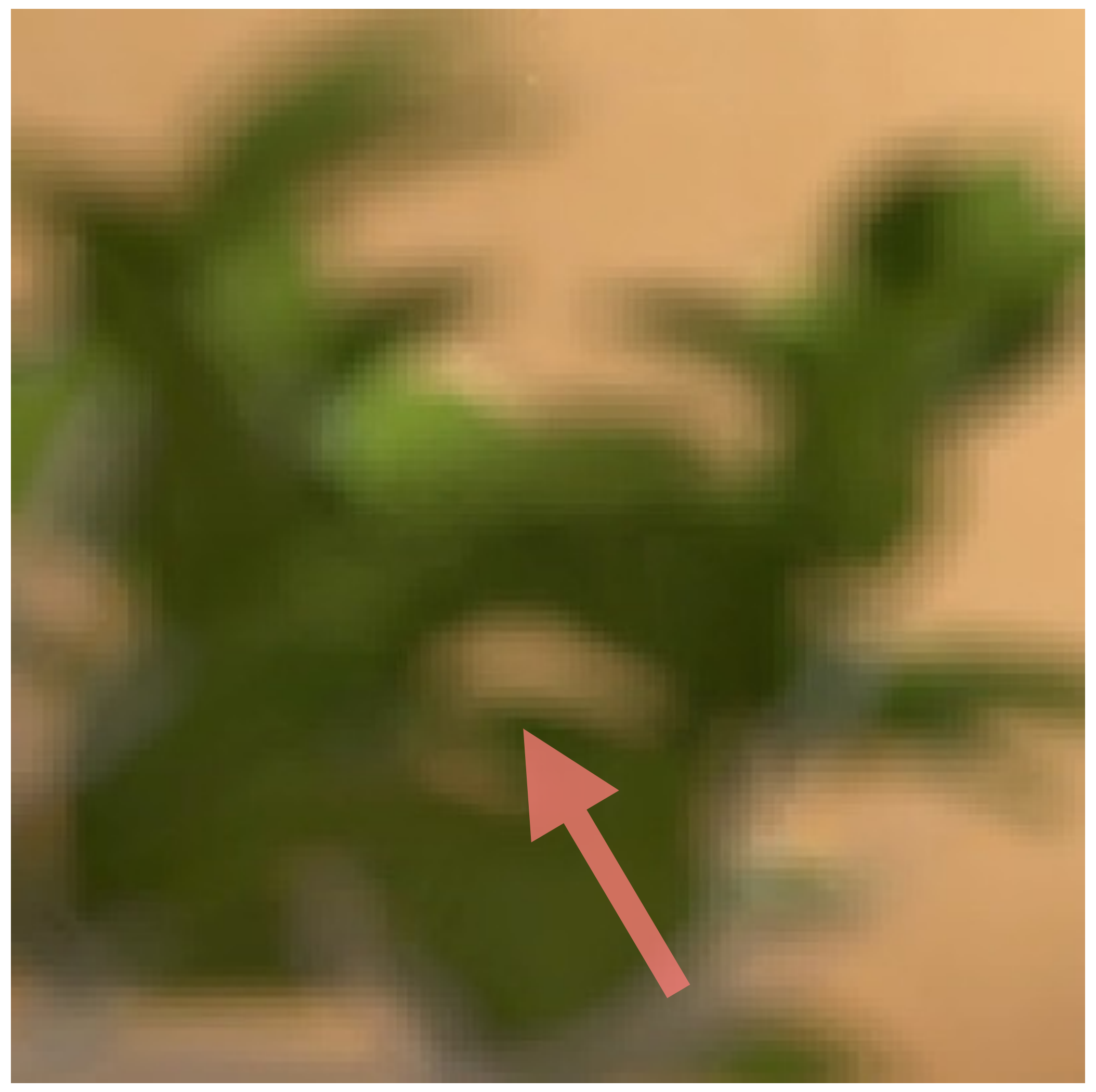
Irradiance Factorization and Depth-of-Field

Input (512 spp)



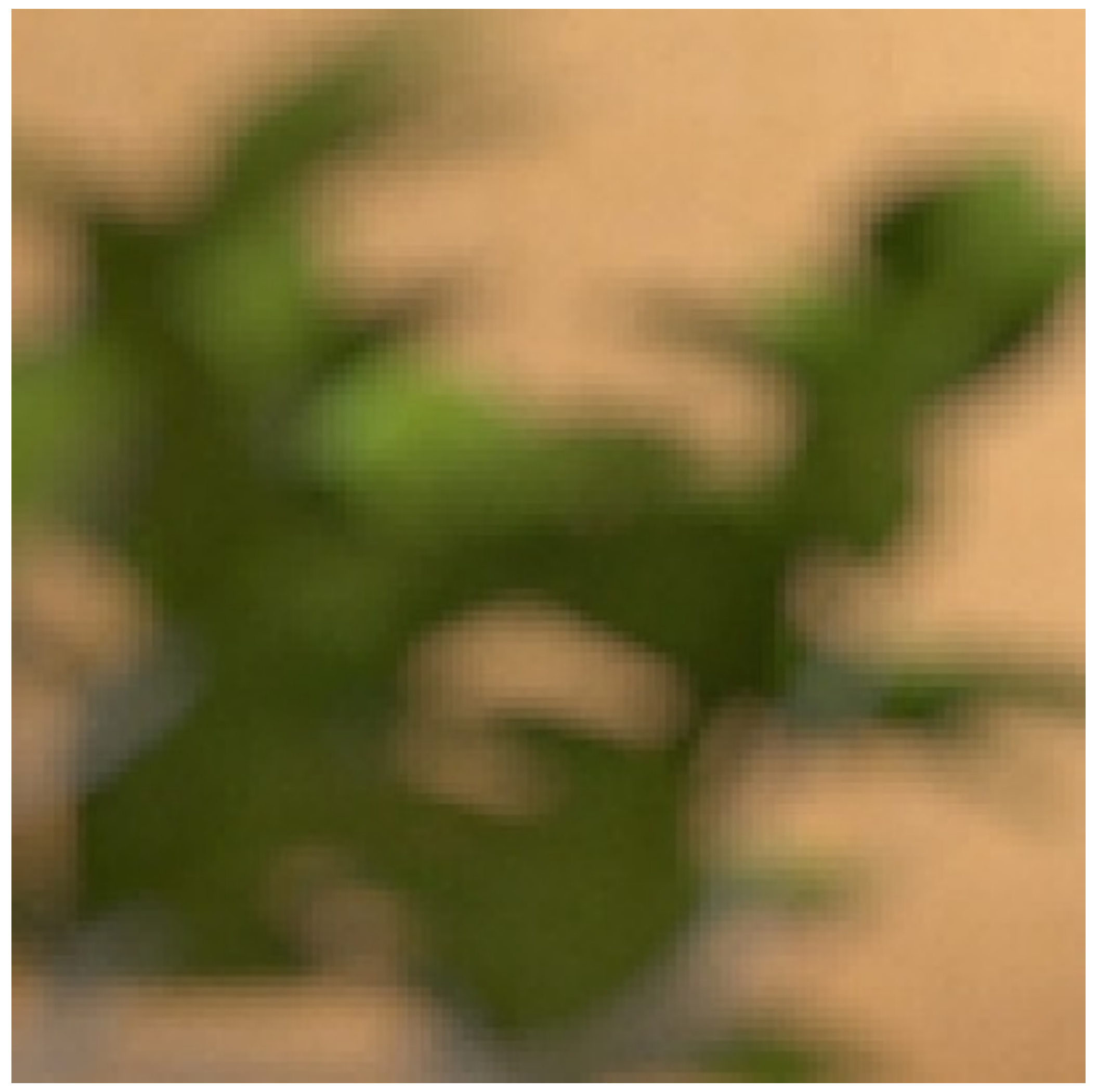
rel. MSE: $5.6 \cdot 10^{-3}$

Std. irradiance

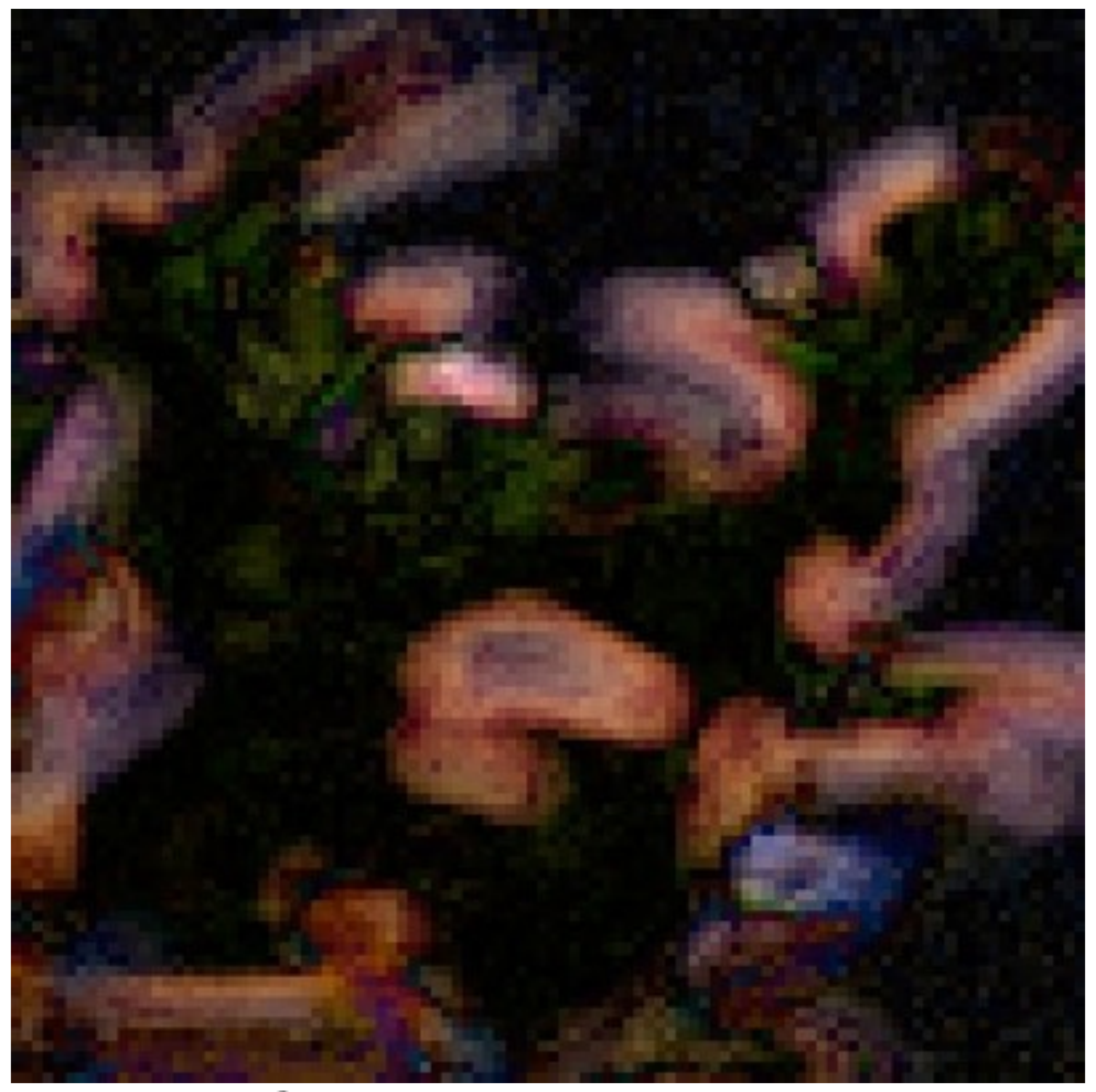


$9.9 \cdot 10^{-3}$

Ground truth



rel. sqr. error



Monday, June 29, 15

To illustrate the robustness of our effective irradiance factorization, we added depth-of-field to our robot scene and then ran our denoising filter. The result shown here is a crop of the bonsai.

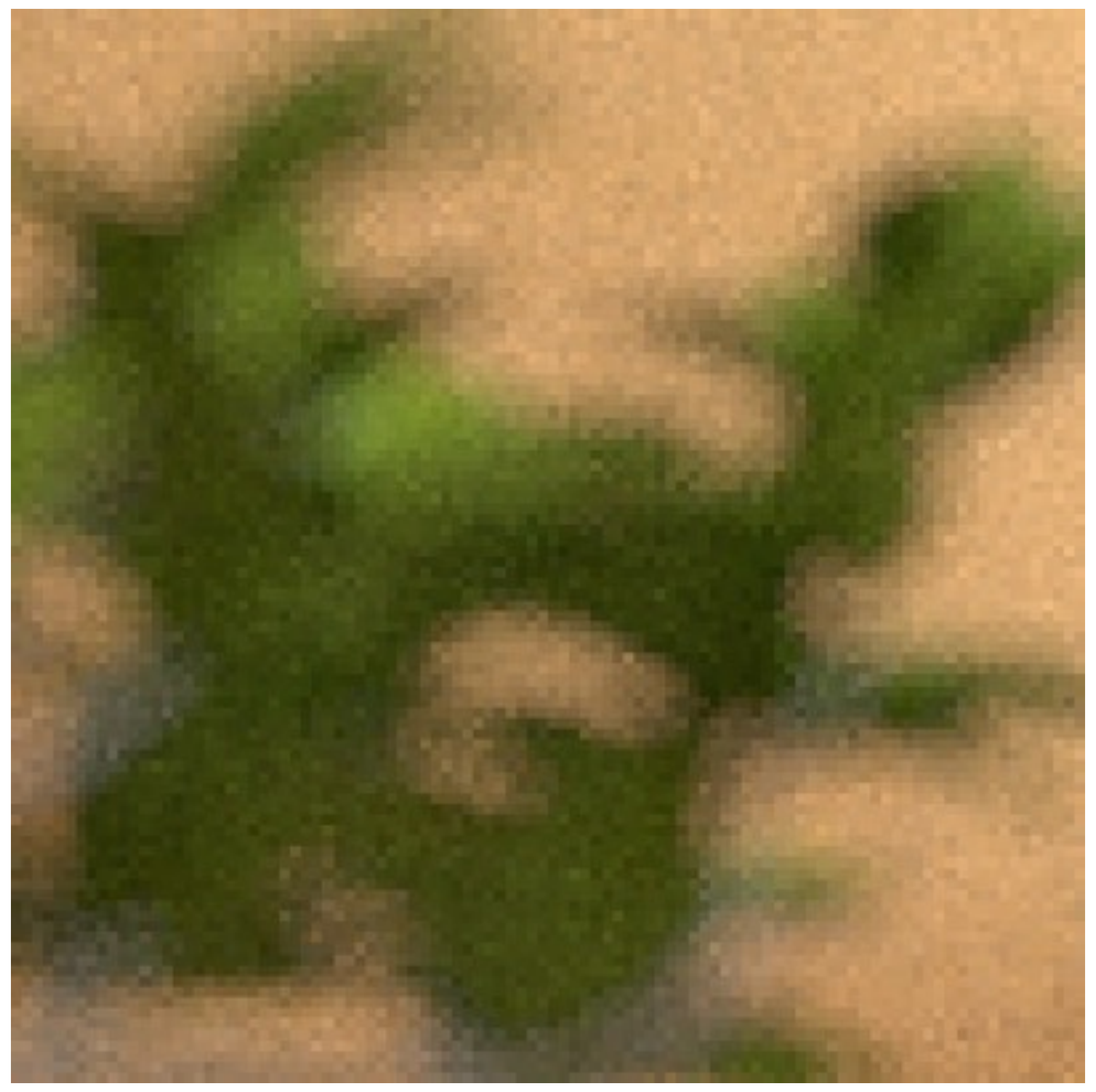
Even though both the bonsai and the sofa in the background are Lambertian surfaces, the depth-of-field effect invalidates the assumption of a linear shading model that standard irradiance factorization relies upon.

While the standard irradiance factorization correctly handles uniform regions, regions with mixed reflectances, such as the one pointed by the arrow, suffer from significant bias. This bias is clearly visible in the bottom row, where we directly visualize the relative squared error, and explains why the filtered output has a higher MSE than the noisy input.

In contrast, our effective irradiance factorization robustly handles regions with mixed reflectances and yields a much better reconstruction overall.

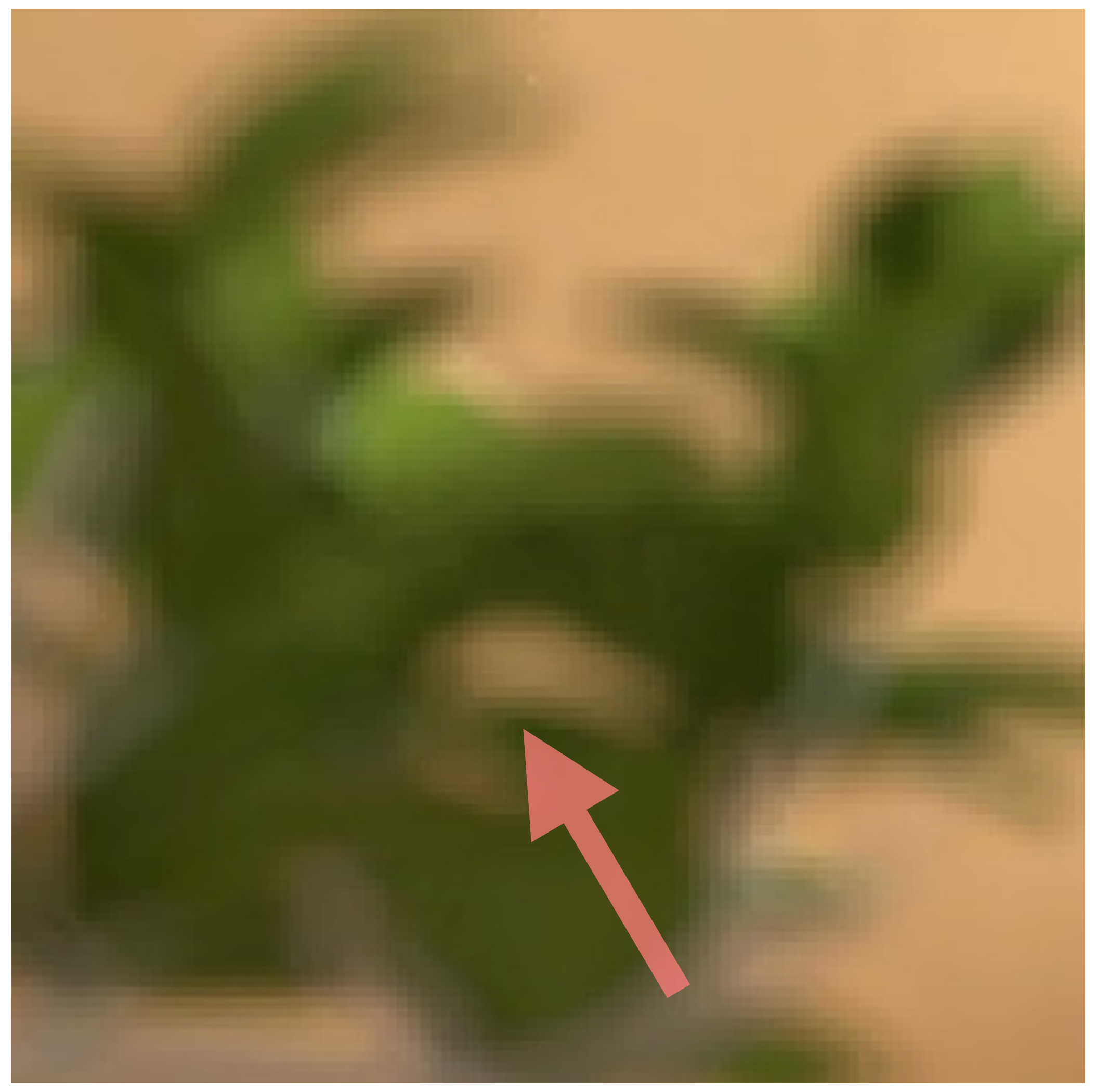
Irradiance Factorization and Depth-of-Field

Input (512 spp)



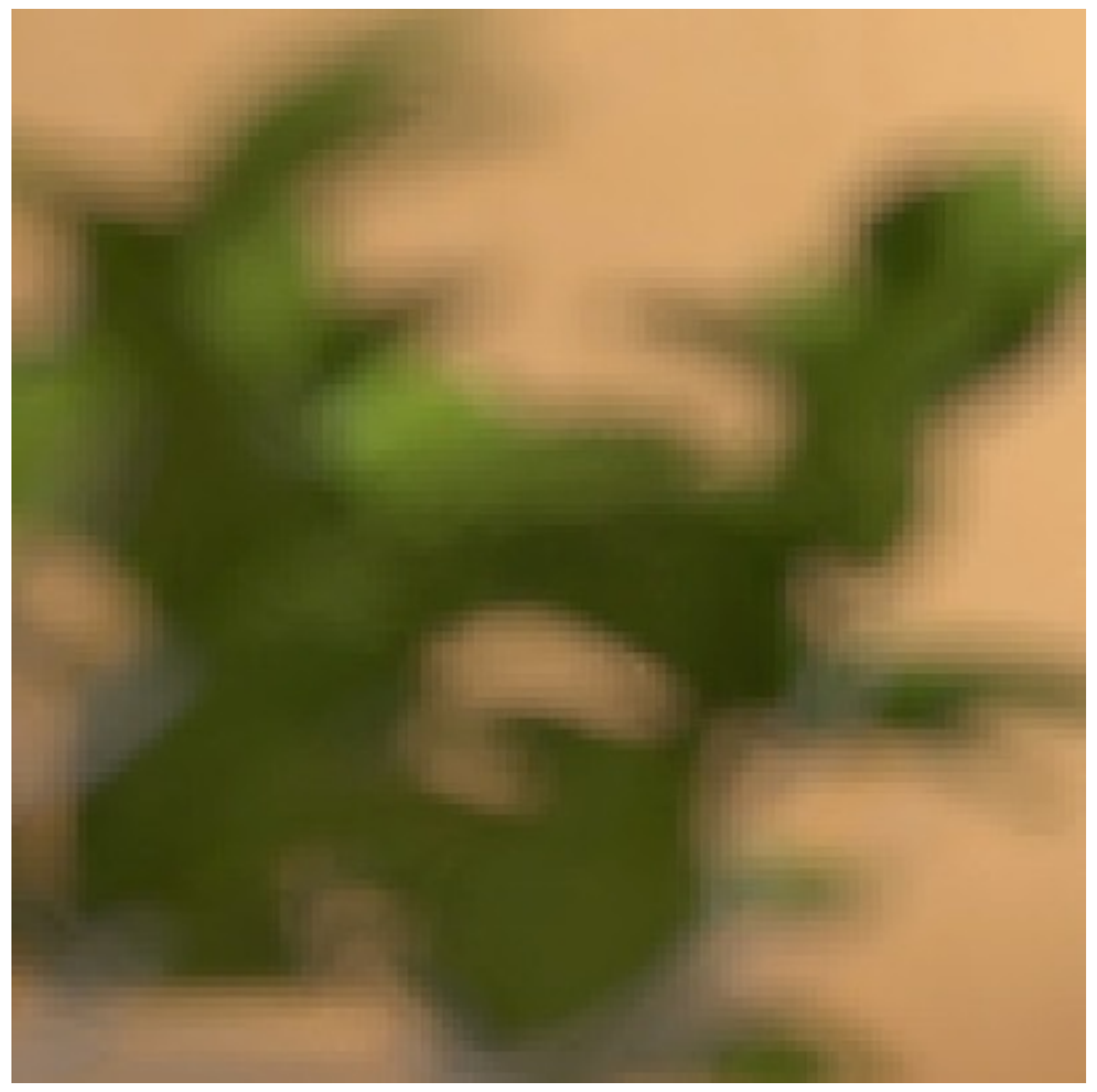
rel. MSE: $5.6 \cdot 10^{-3}$

Std. irradiance



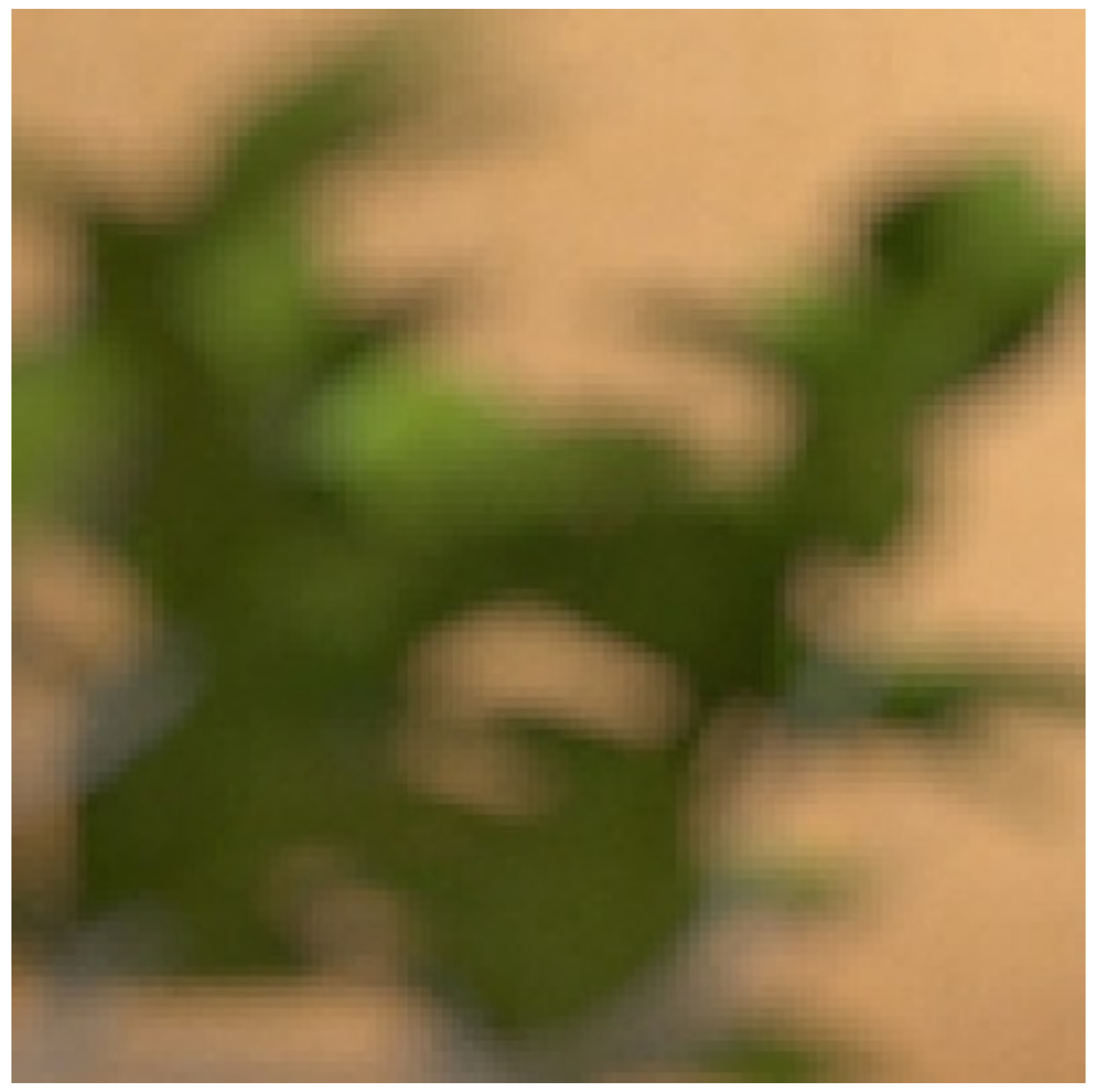
$9.9 \cdot 10^{-3}$

Eff. irradiance

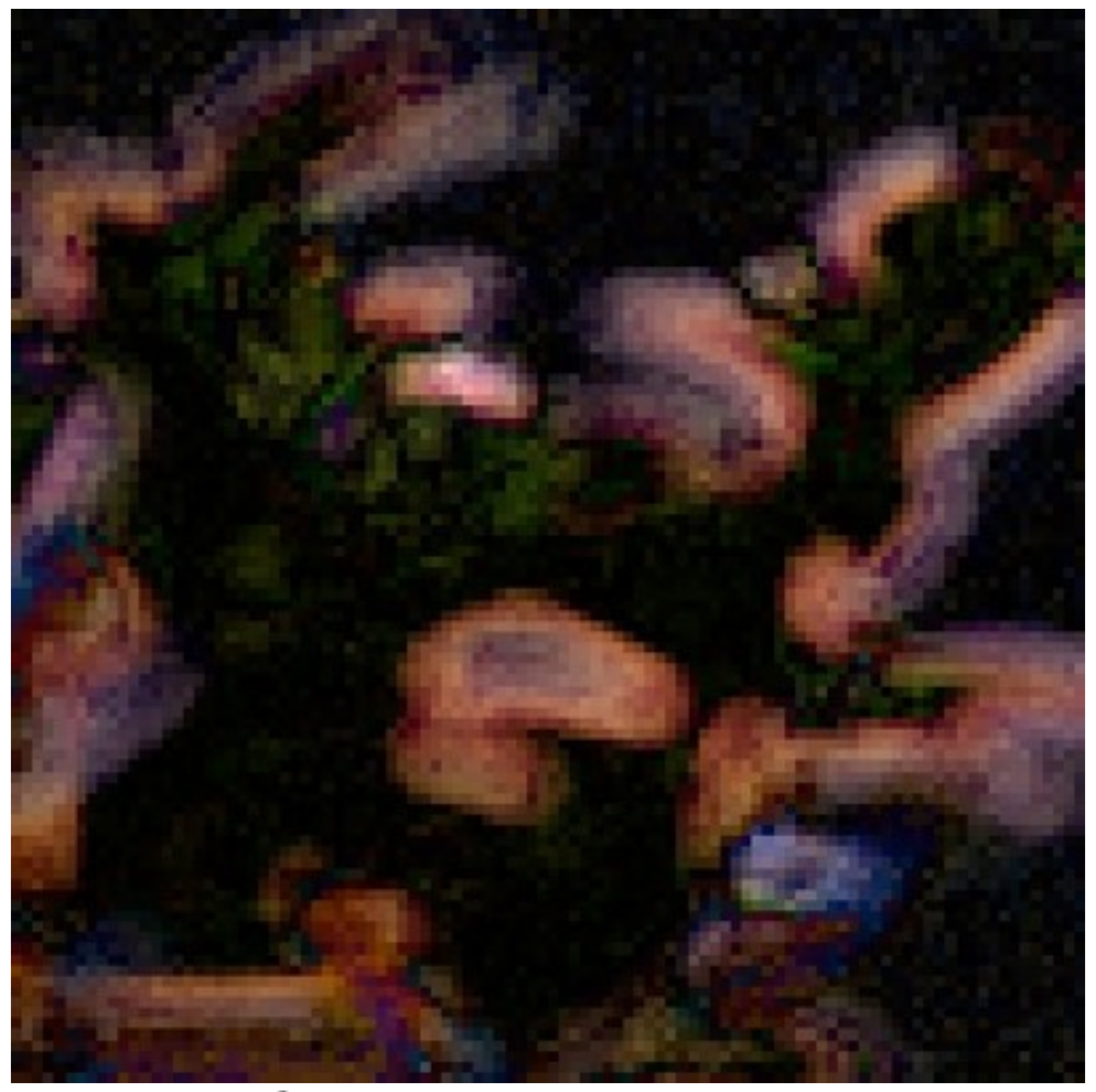


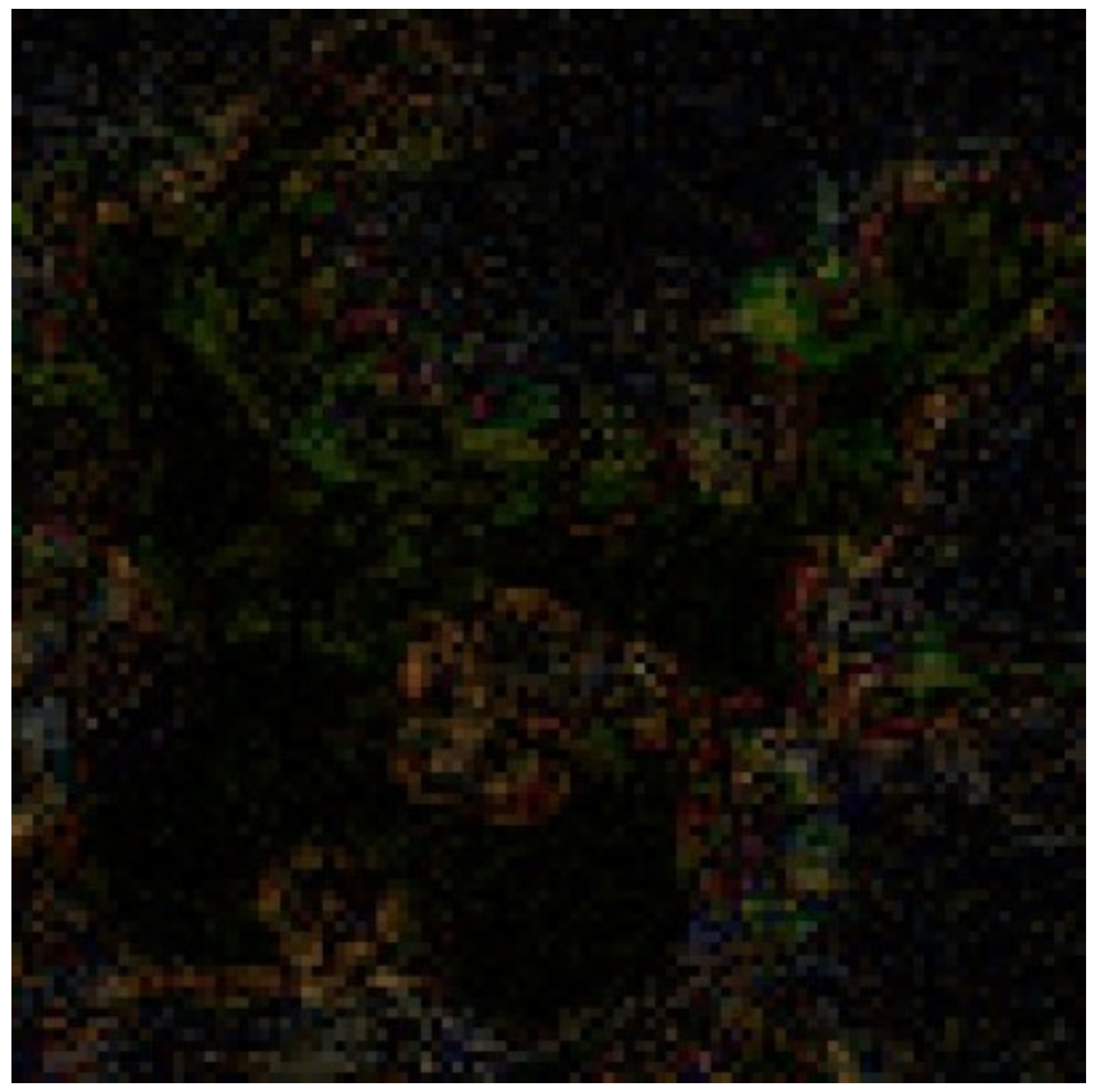
$0.90 \cdot 10^{-3}$

Ground truth



rel. sqr. error





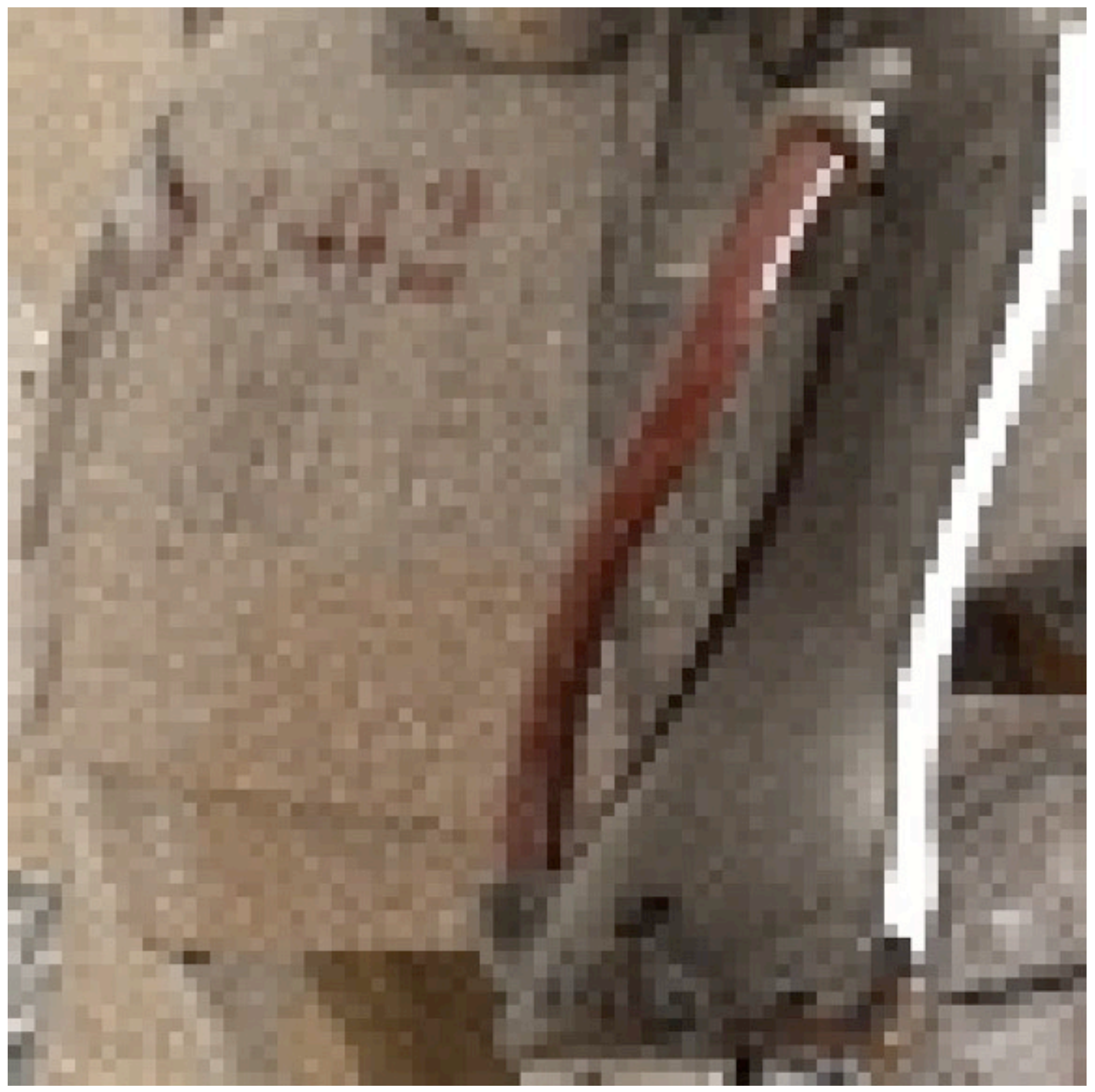
To illustrate the robustness of our effective irradiance factorization, we added depth-of-field to our robot scene and then ran our denoising filter. The result shown here is a crop of the bonsai.

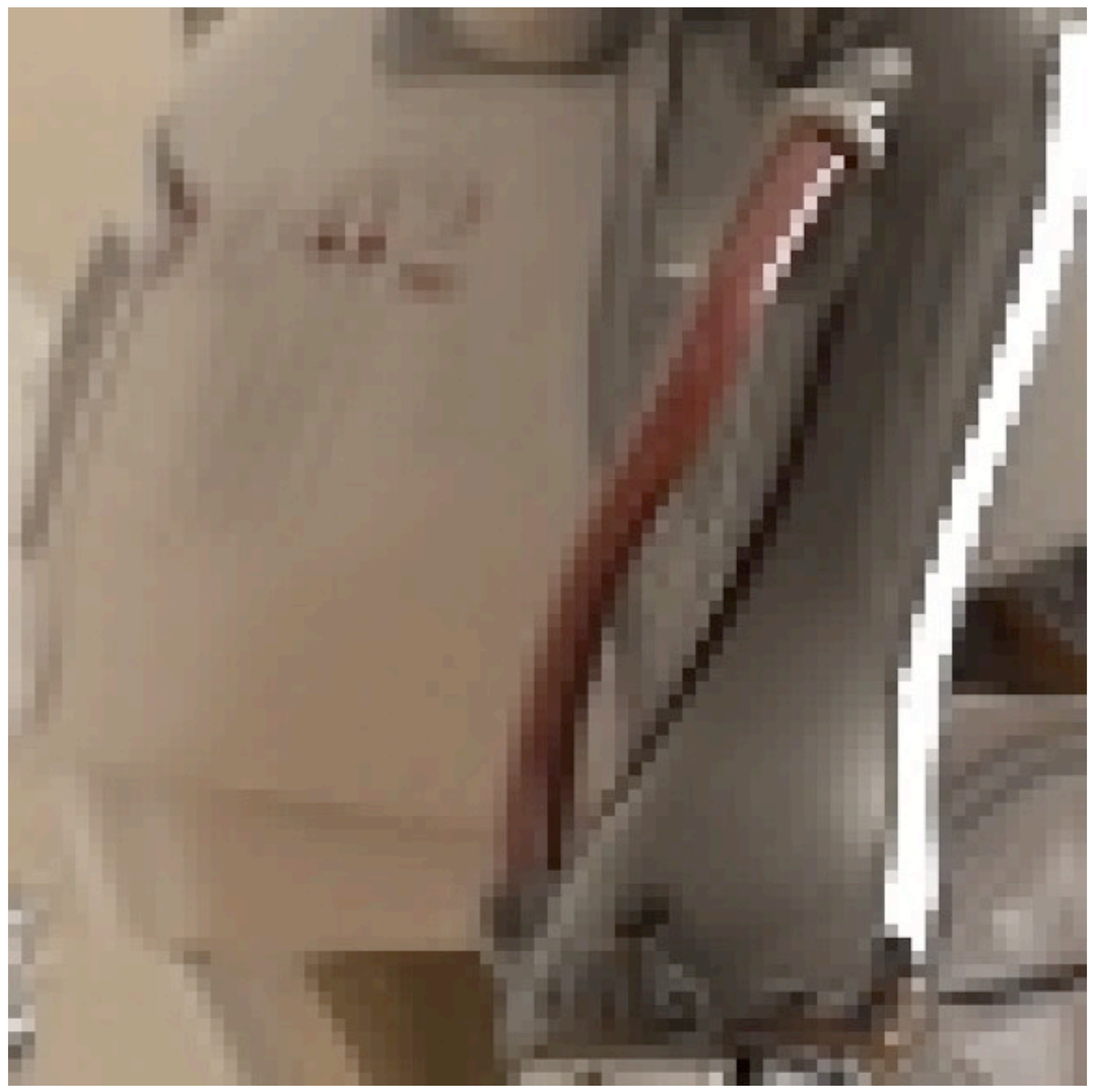
Even though both the bonsai and the sofa in the background are Lambertian surfaces, the depth-of-field effect invalidates the assumption of a linear shading model that standard irradiance factorization relies upon.

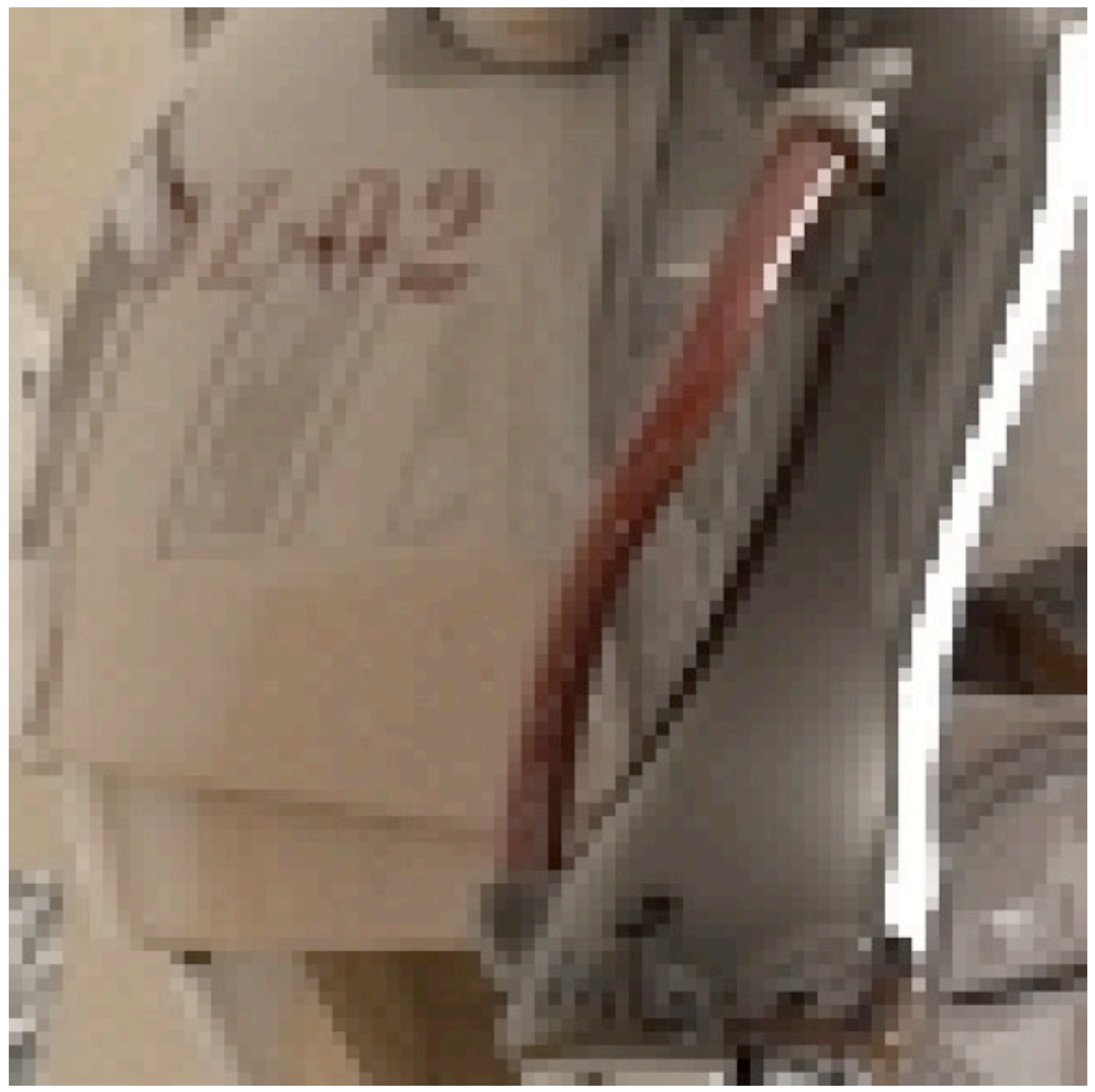
While the standard irradiance factorization correctly handles uniform regions, regions with mixed reflectances, such as the one pointed by the arrow, suffer from significant bias. This bias is clearly visible in the bottom row, where we directly visualize the relative squared error, and explains why the filtered output has a higher MSE than the noisy input.

In contrast, our effective irradiance factorization robustly handles regions with mixed reflectances and yields a much better reconstruction overall.

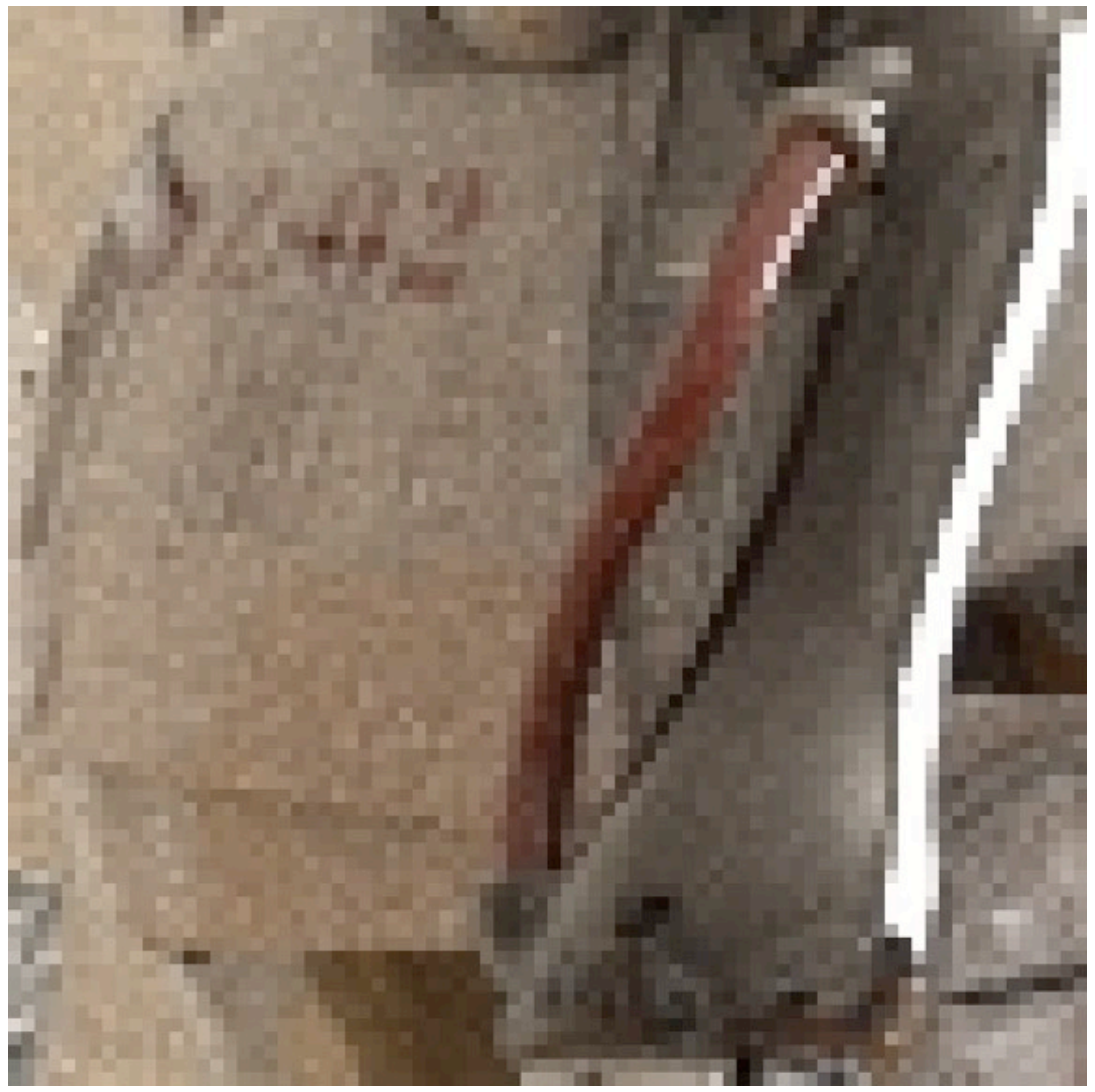
Denoising Results

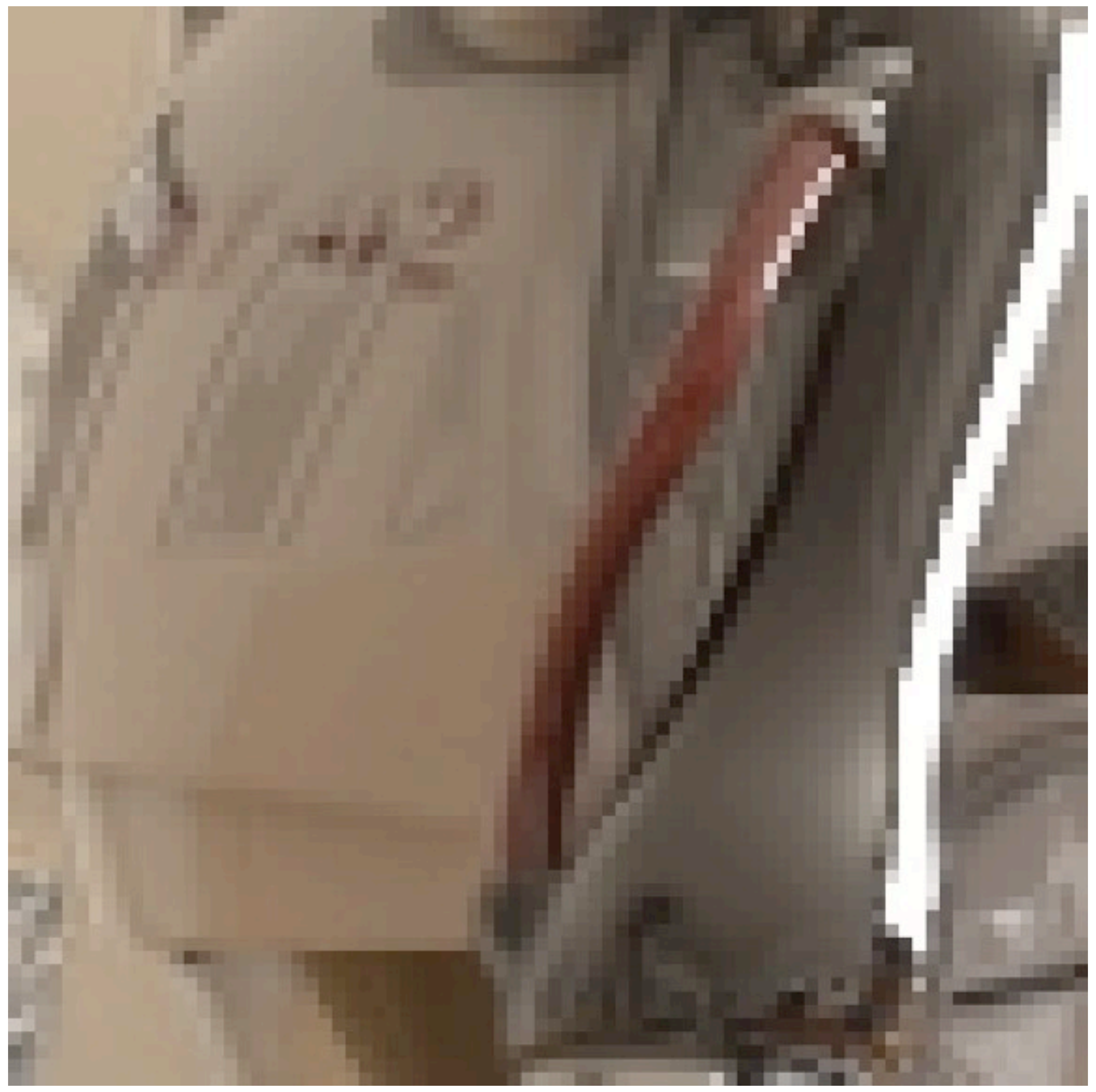

Input (512 spp)

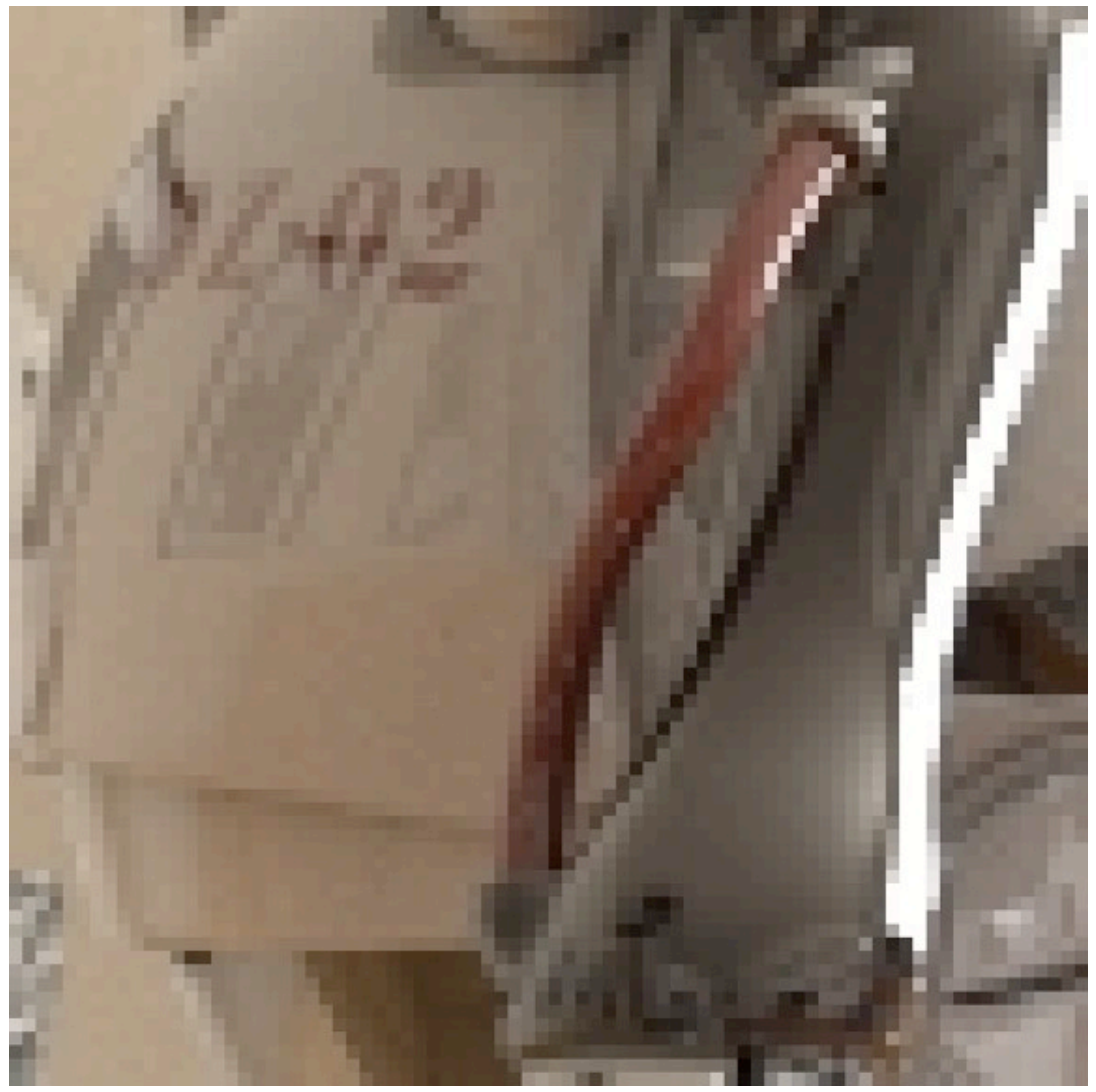

NL-Means


Ground truth (16k spp)

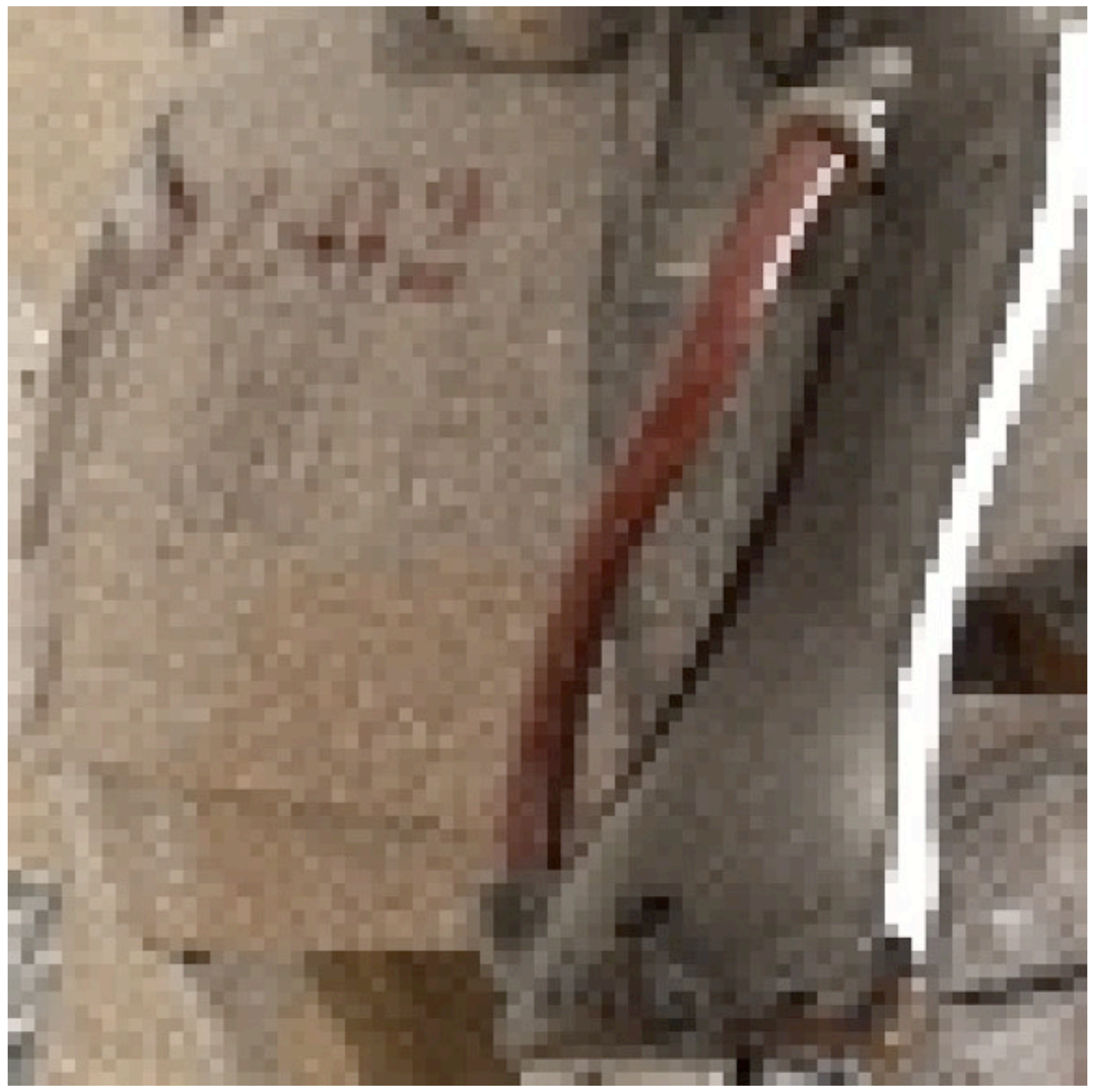
Denoising Results

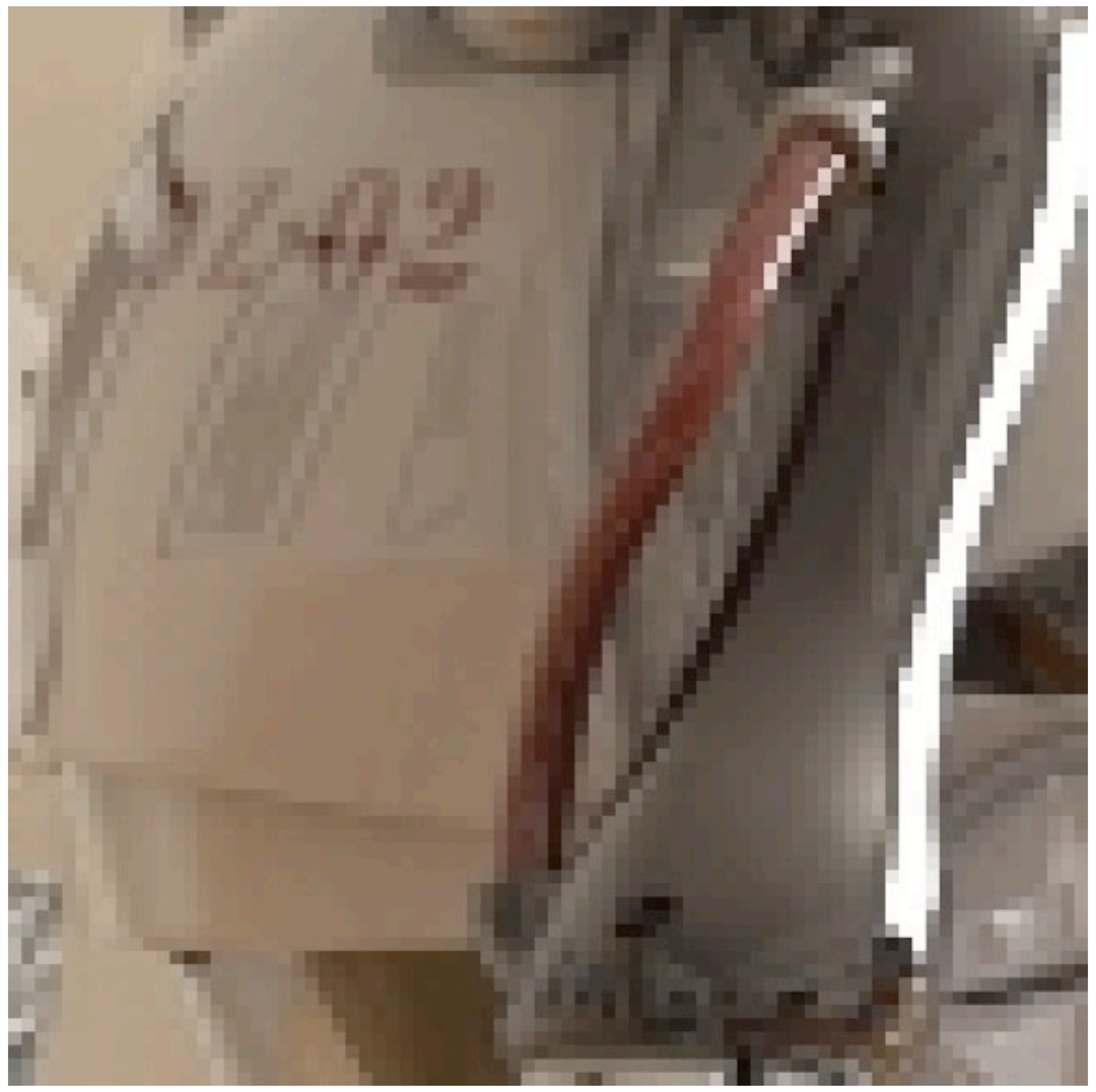

Input (512 spp)

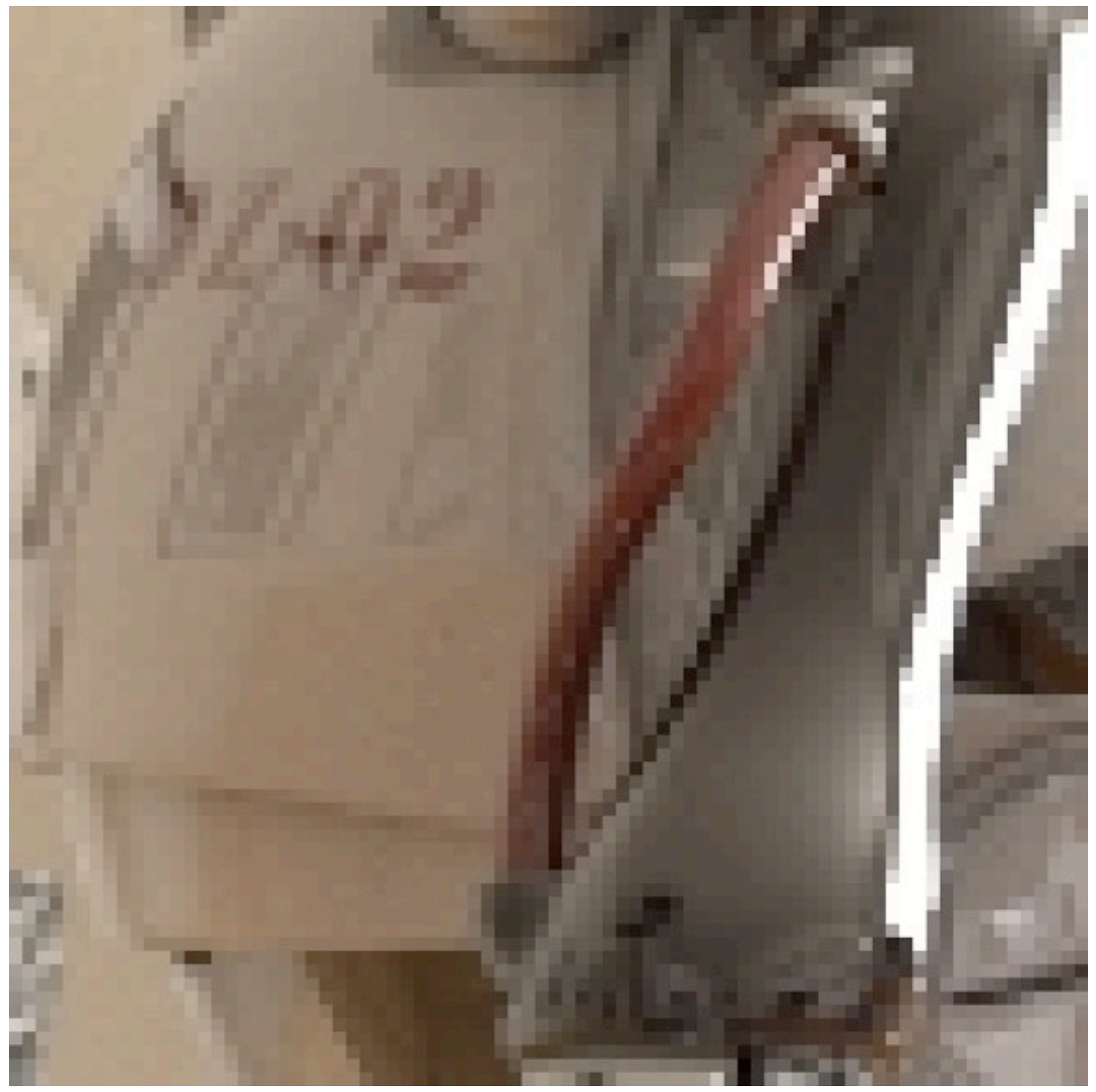

NL-Means
+ decomposition


Ground truth (16k spp)

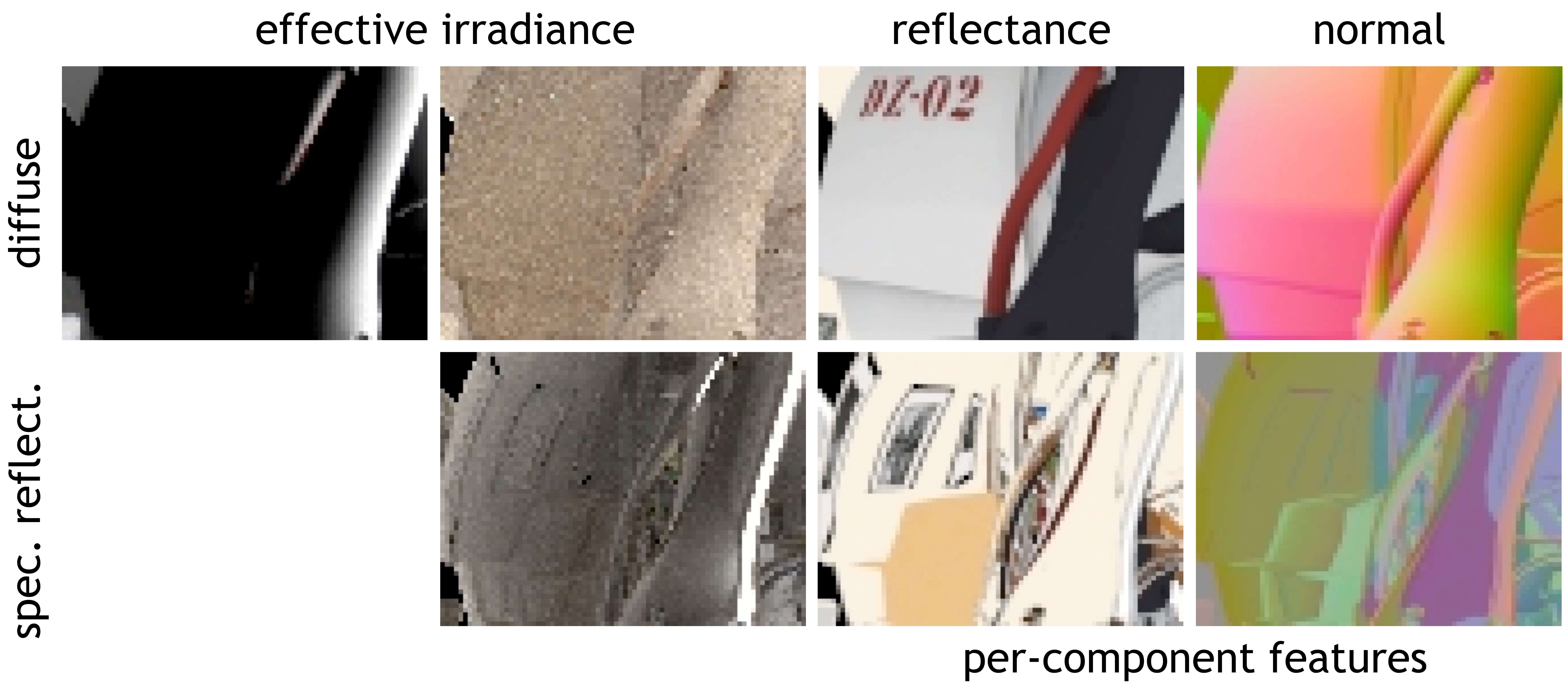
Denoising Results


Input (512 spp)

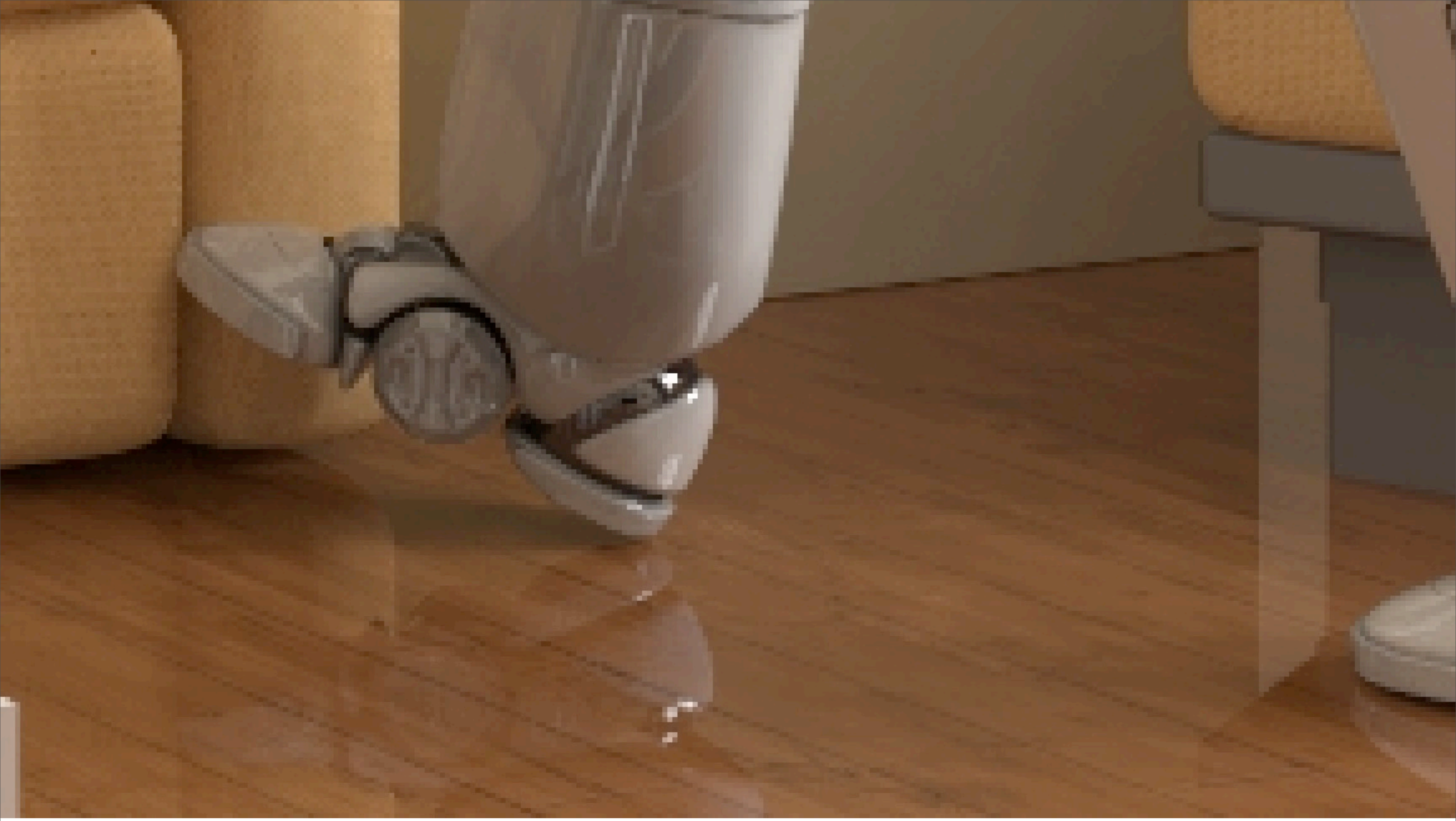

NL-Means
+ decomposition
+ features
(ours)


Ground truth (16k spp)

Denoising Results



With irradiance factorization, rMSE: $0.77 \cdot 10^{-3}$



Without irradiance factorization, rMSE: $0.98 \cdot 10^{-3}{}_{64}$

Performance Details

relative cost = overhead wrt. baseline / resolution gain

	Overhead	Resolution	Relative cost
Baseline (512 spp)	1	1x	**1**
Reference (16k spp)	32	1x	**32**
Denoising	1.2	1x	**1.2**
D + Interpolation	2.2	4x	**0.55**
D + Upsampling	1.72	4x	**0.43**
All (D + U + I)	4.64	16x	**0.29**

Memory overhead

proportional to decomposition granularity

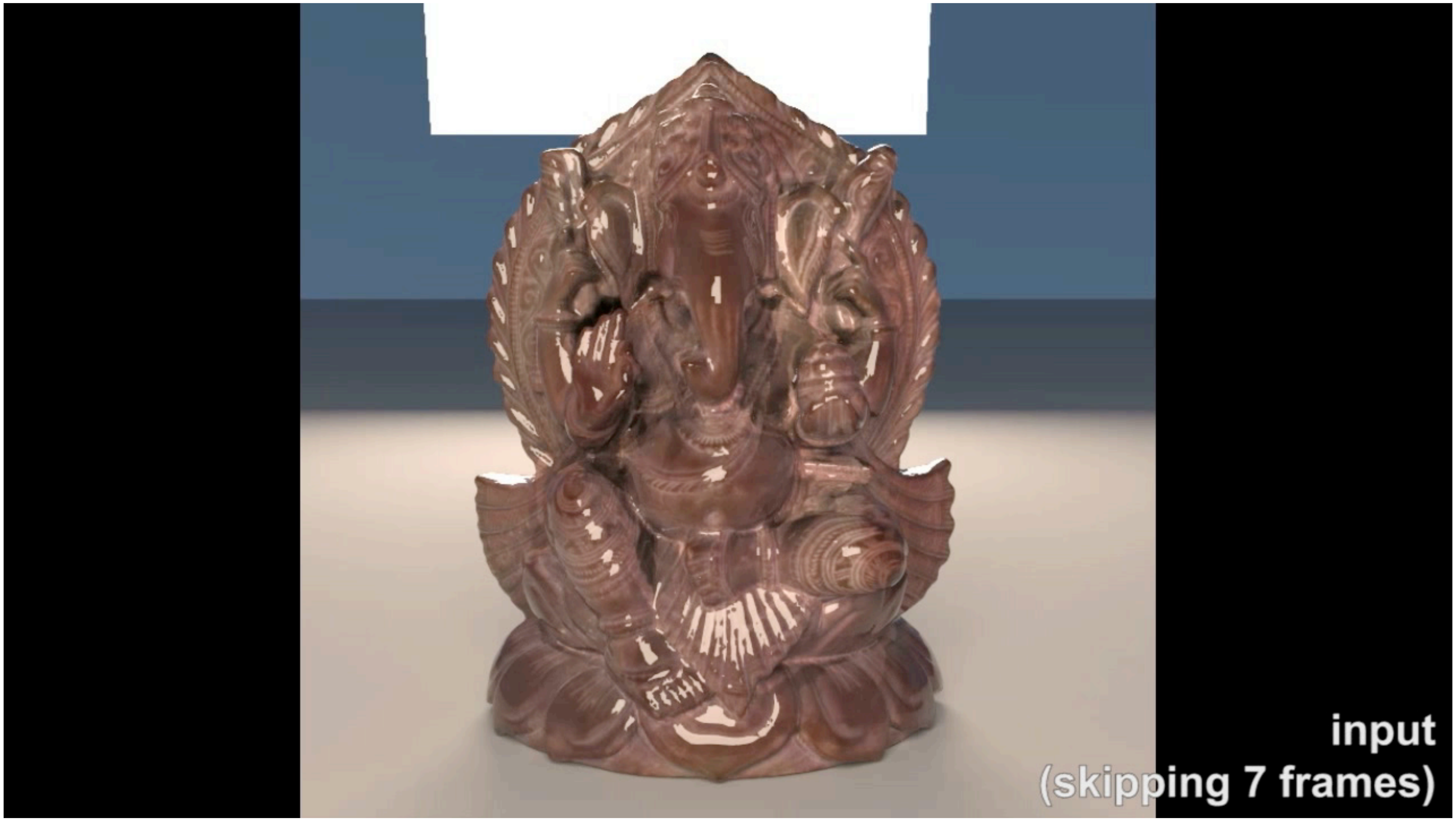
store as compressed multi-channel EXRs

Example (Robot)

full decomposition:	29.0 MB
final pixel color:	3.7 MB

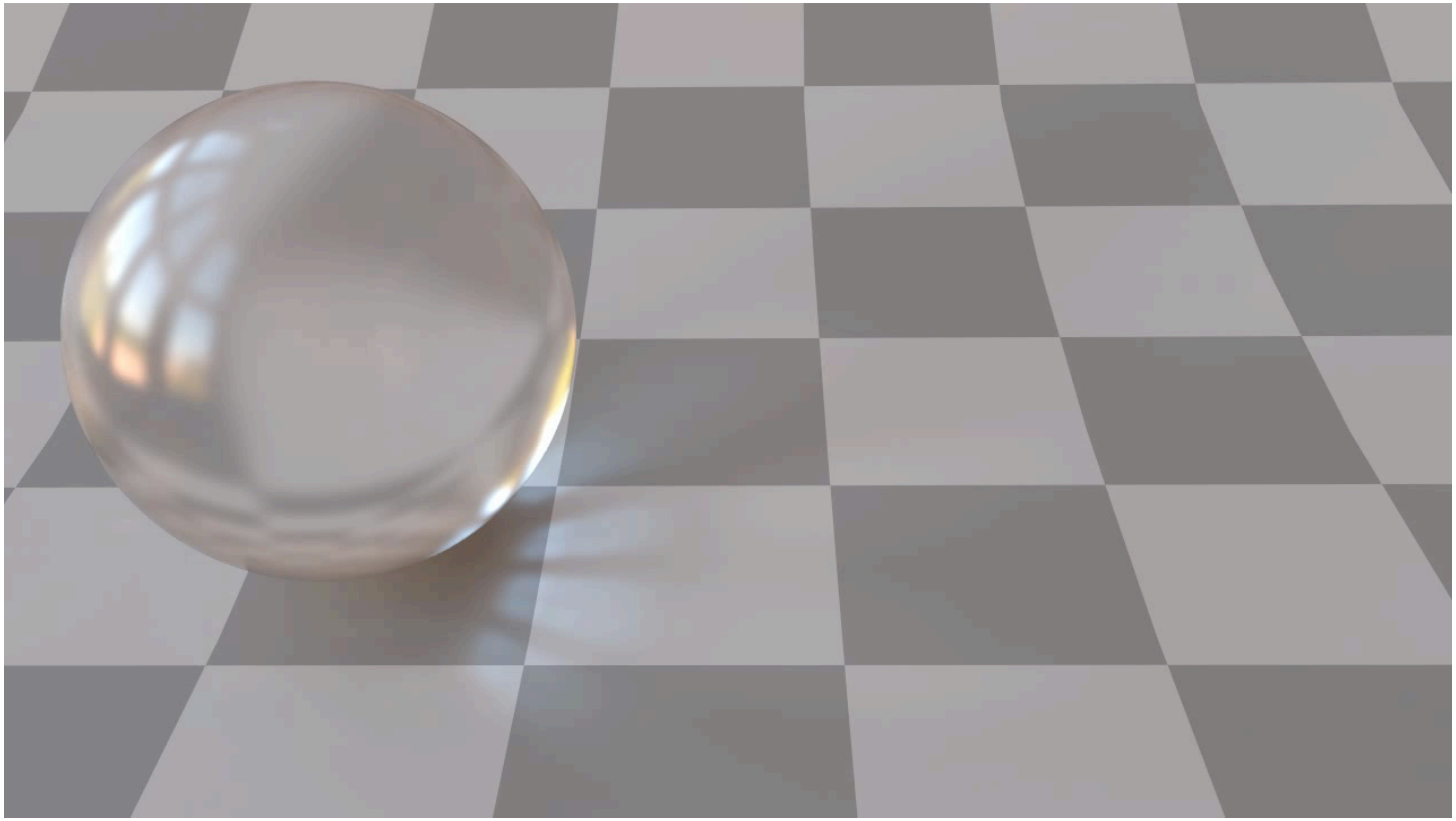
EXRs: leveraging that some components have many zero pixels

More Specular Interpolation

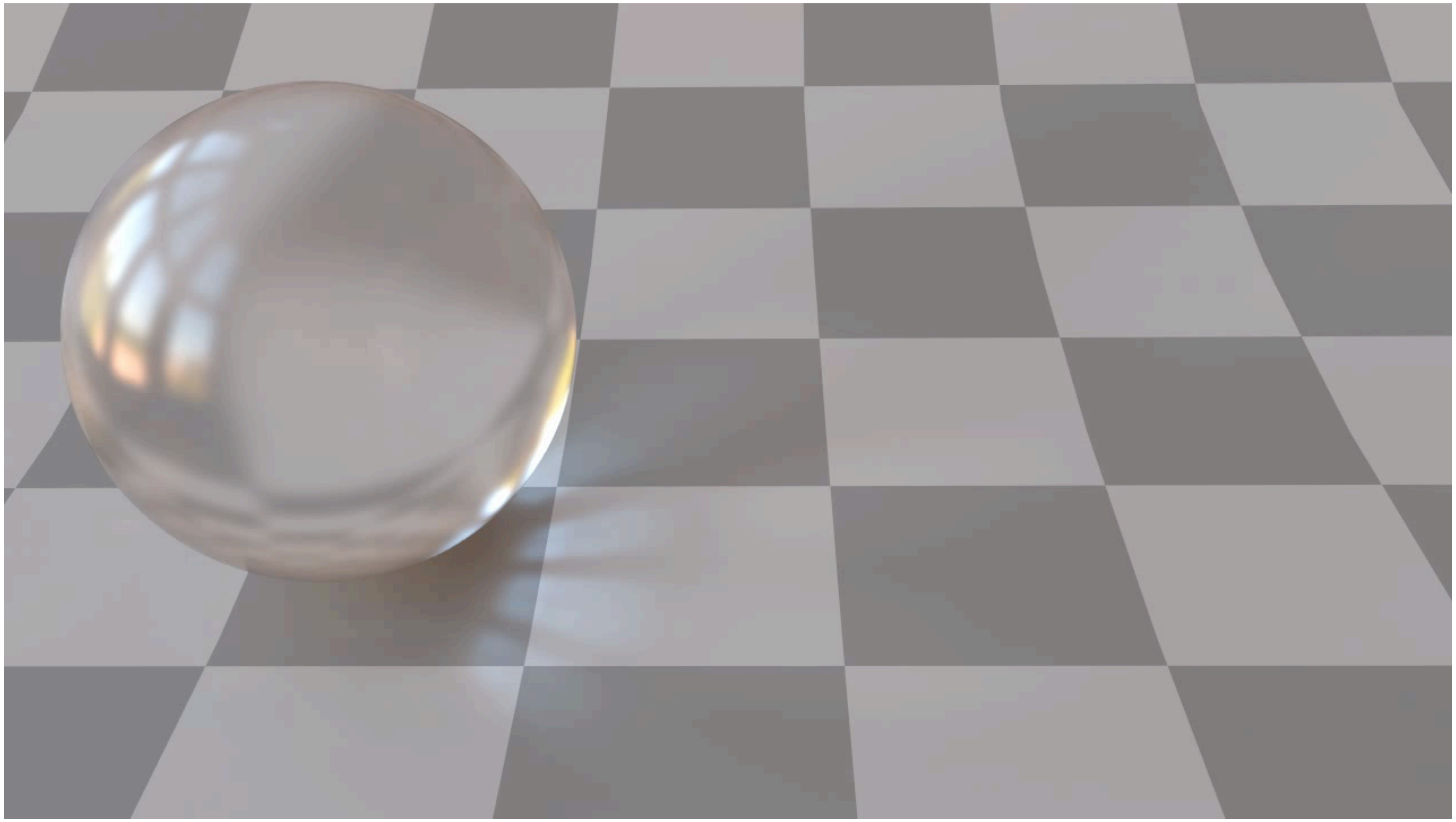


not treating these light paths specially can lead to significant errors in the interpolation, as you can see in this example where a light source is specularly reflected on a 3d-scanned statue. To achieve a better interpolation, we needed a way of computing the image-space motion of objects seen through specular scattering events.

Glossy Interpolation

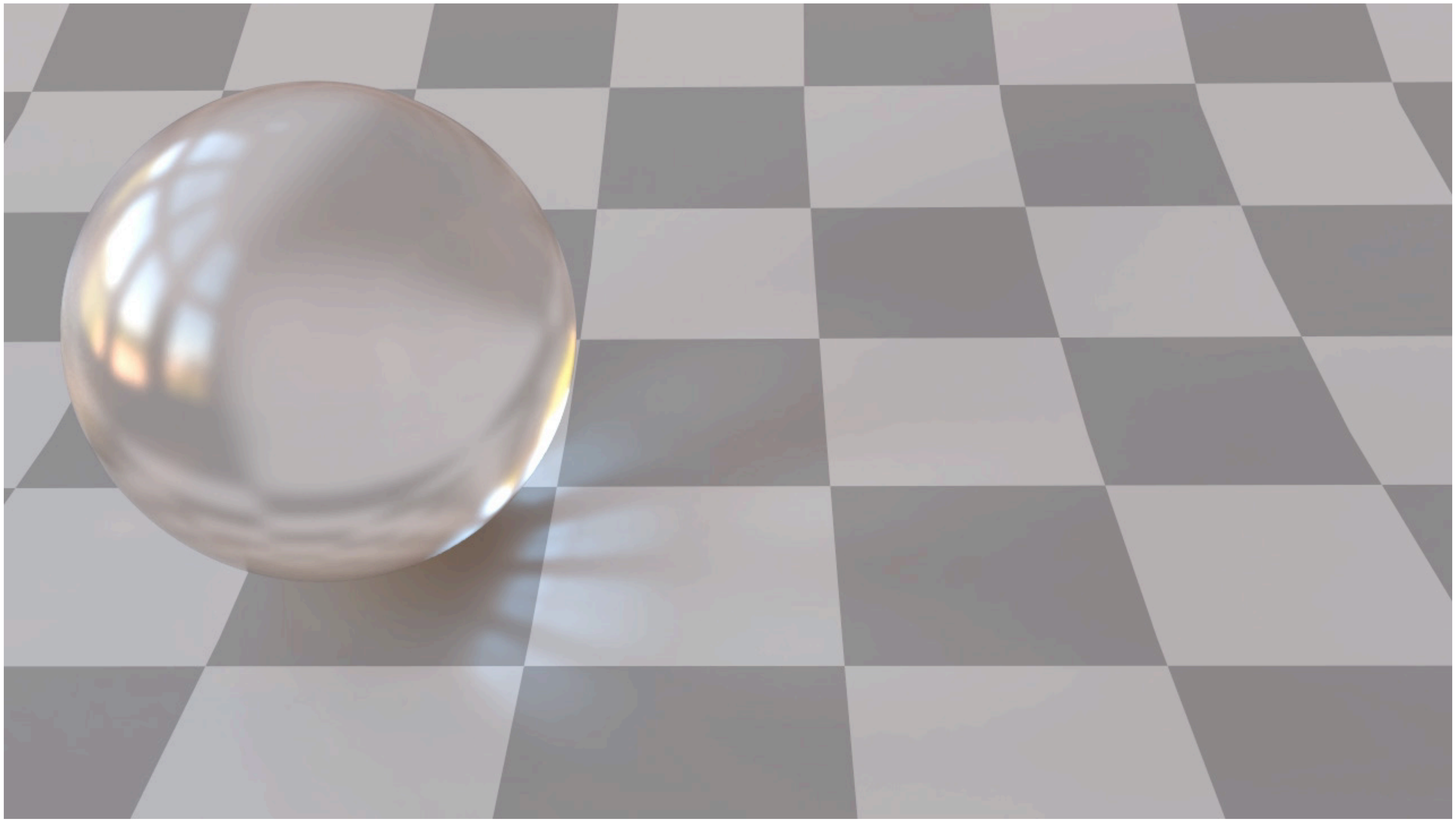

Naive Interpolation

Glossy Interpolation



Our result

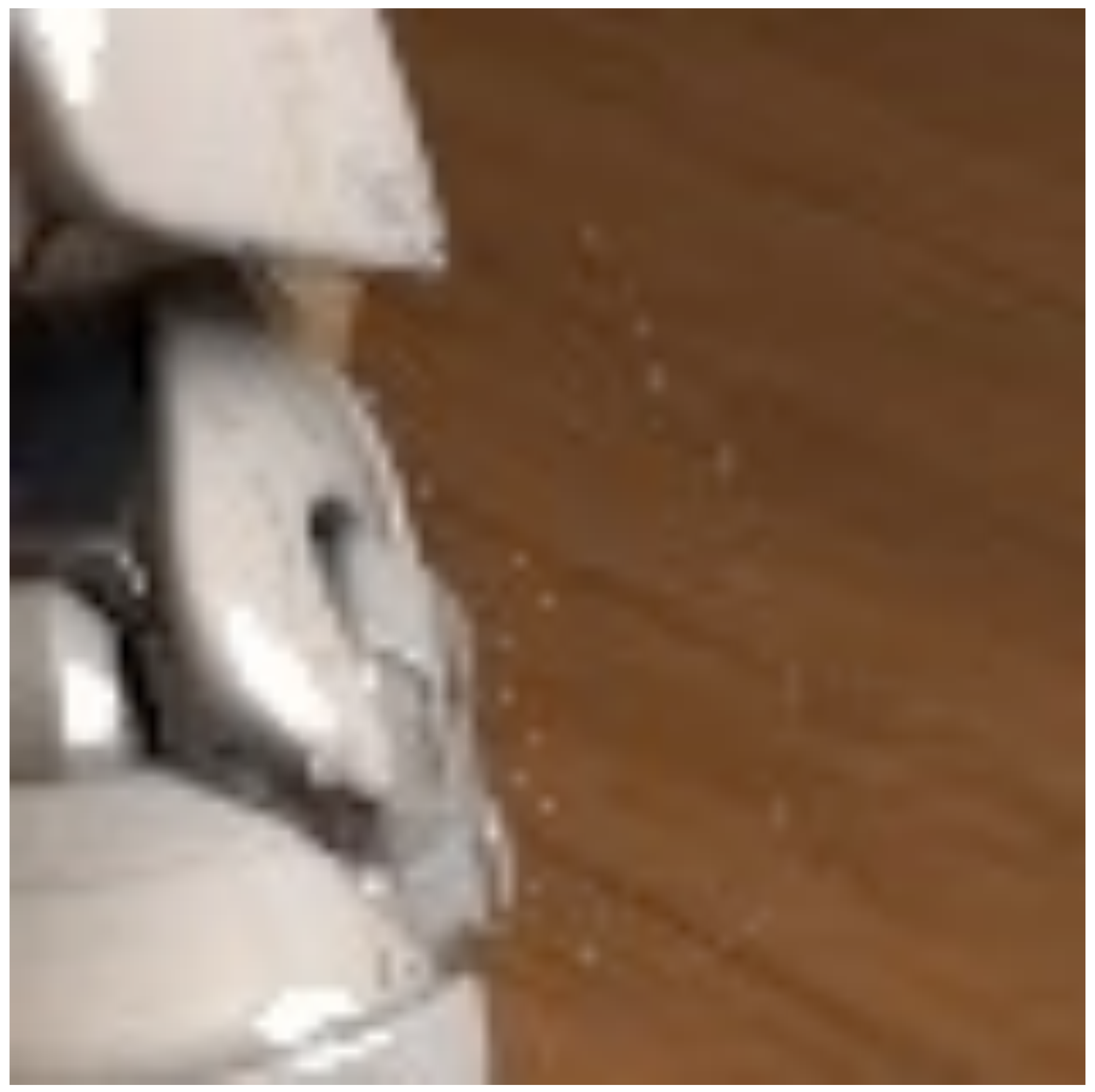
Glossy Interpolation

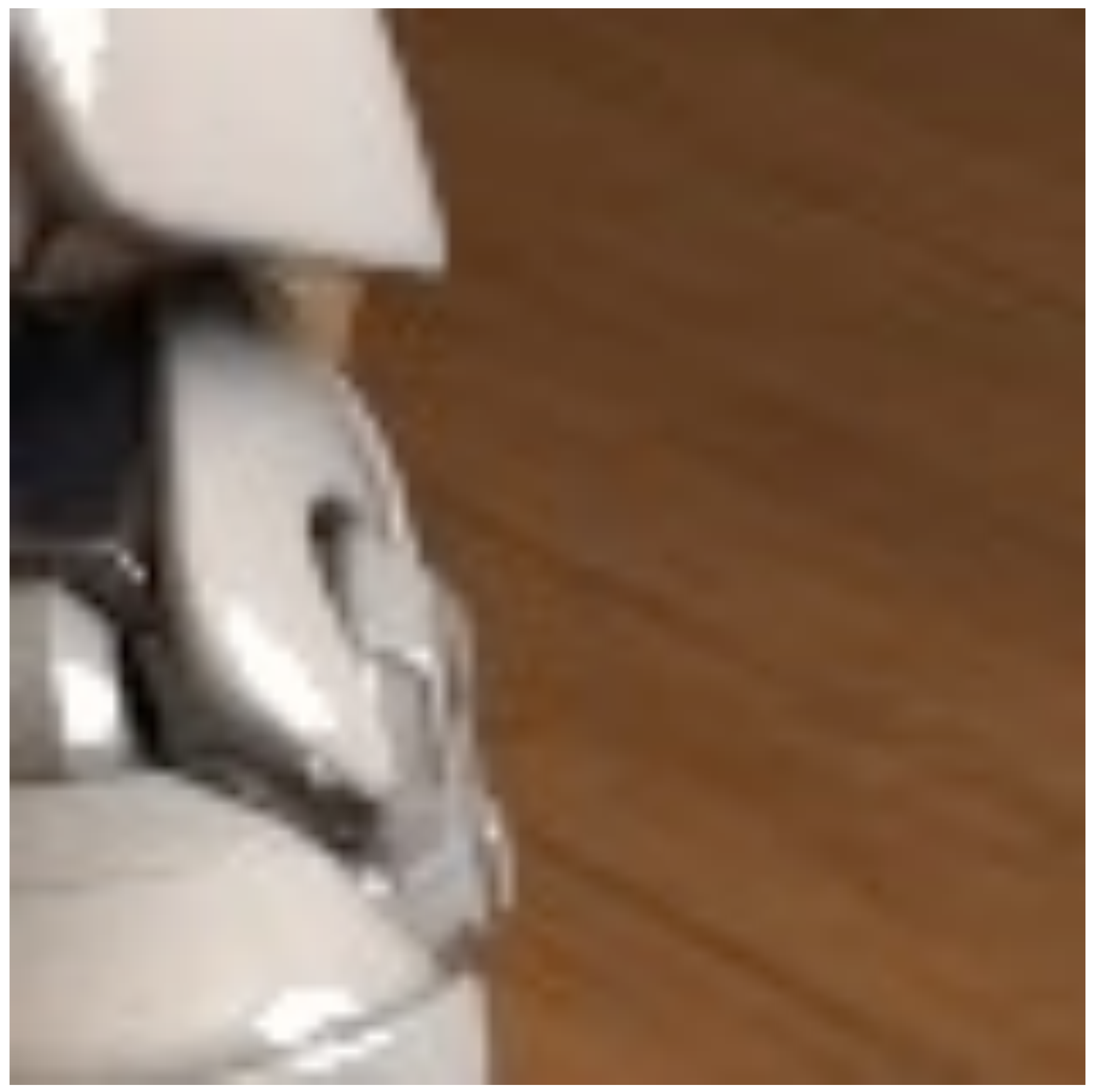

Ground truth

Limitations

Sliver re-rendering for interpolation

Silhouettes


Interpolating silhouette pixels


Re-rendering silhouette pixels

Let us look at the specific problems for silhouettes and occlusions in more detail

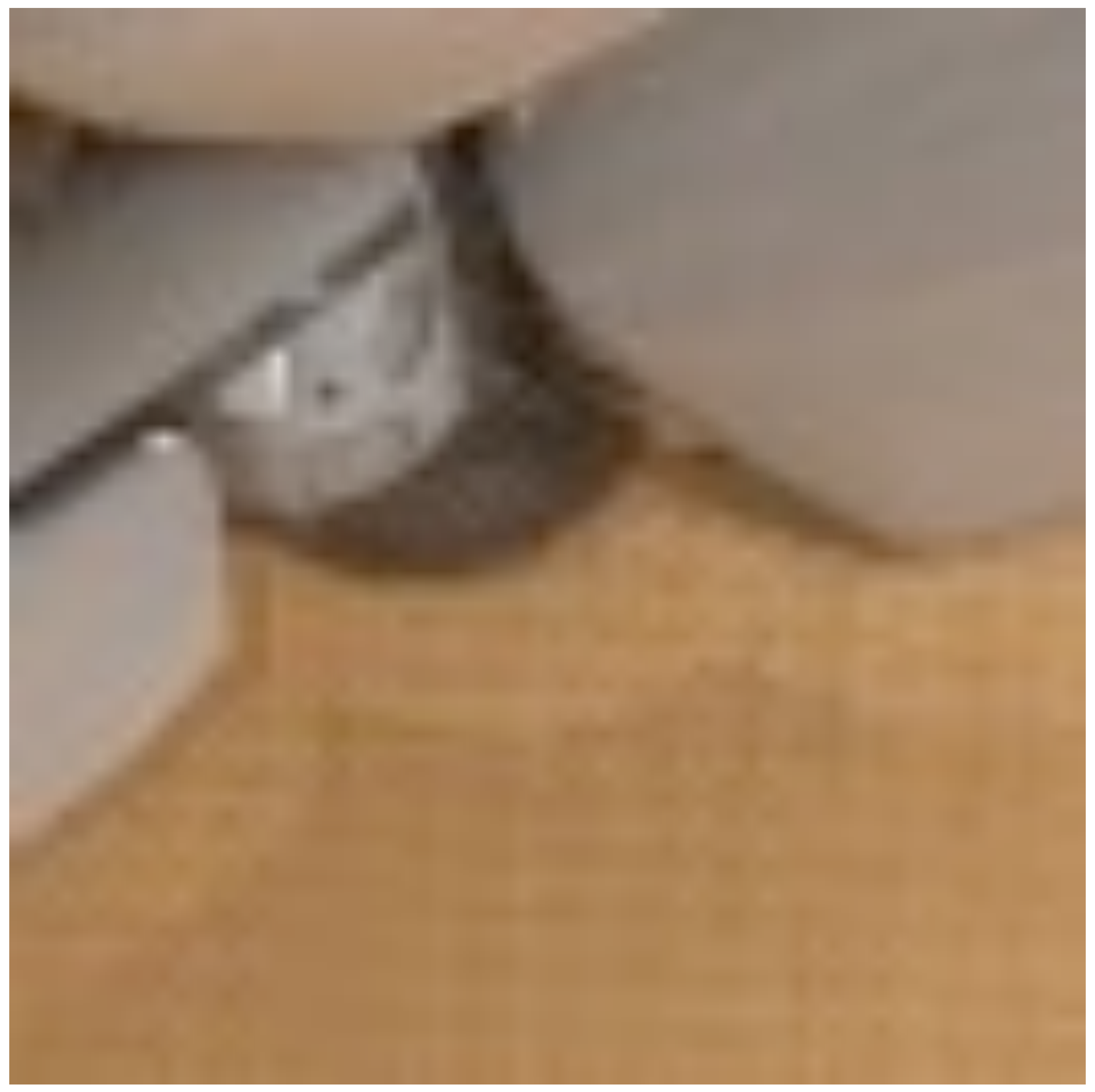
Silhouette pixels capture both fore- and background colors, but they have only a single motion vector assigned to them, that is defined in the center of the pixel.
Consequently, interpolating such silhouette pixels can lead to spurious artifacts as shown on the left.

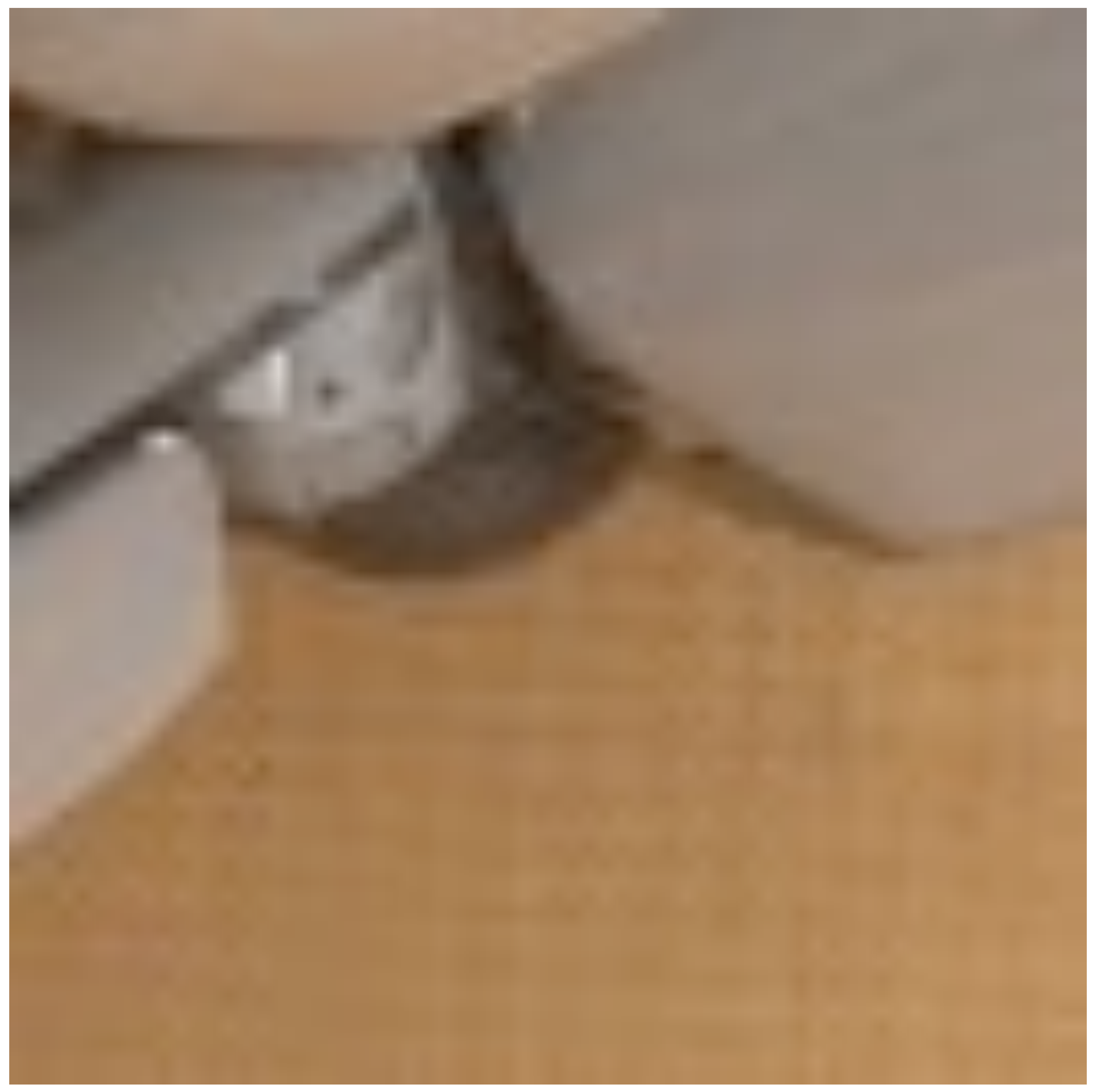
<SCRATCH???>

Limitations

Sliver re-rendering for interpolation

- Silhouettes
- Occlusions


Without re-rendering


Re-rendering occlusions

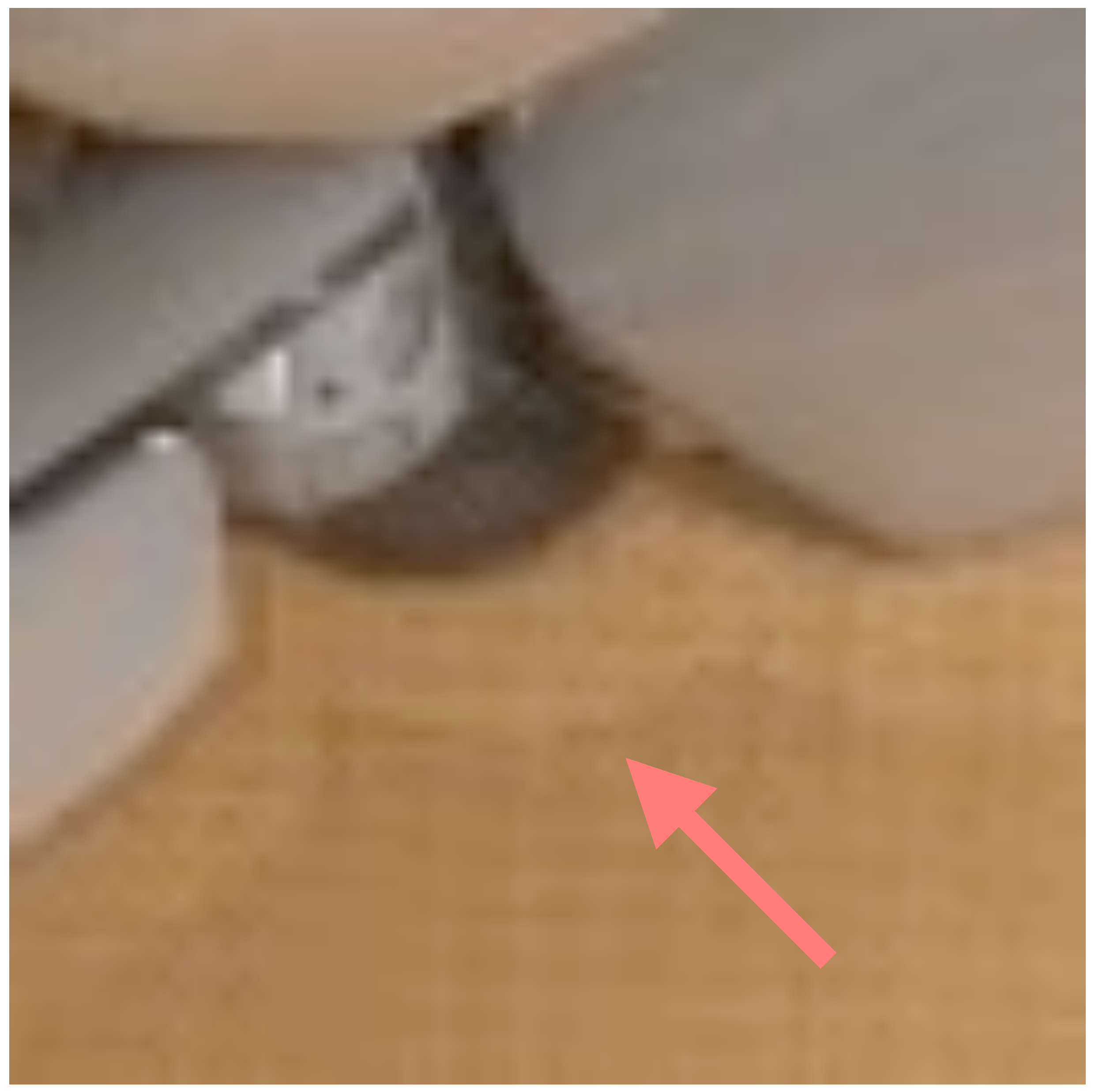
For interpolation one typically averages contributions from the two neighboring keyframes.
But at occlusions, only a single keyframe contributes and if the illumination changed strongly between the keyframes, a seam at the interface between occluded and un-occluded regions will appear.
<click>

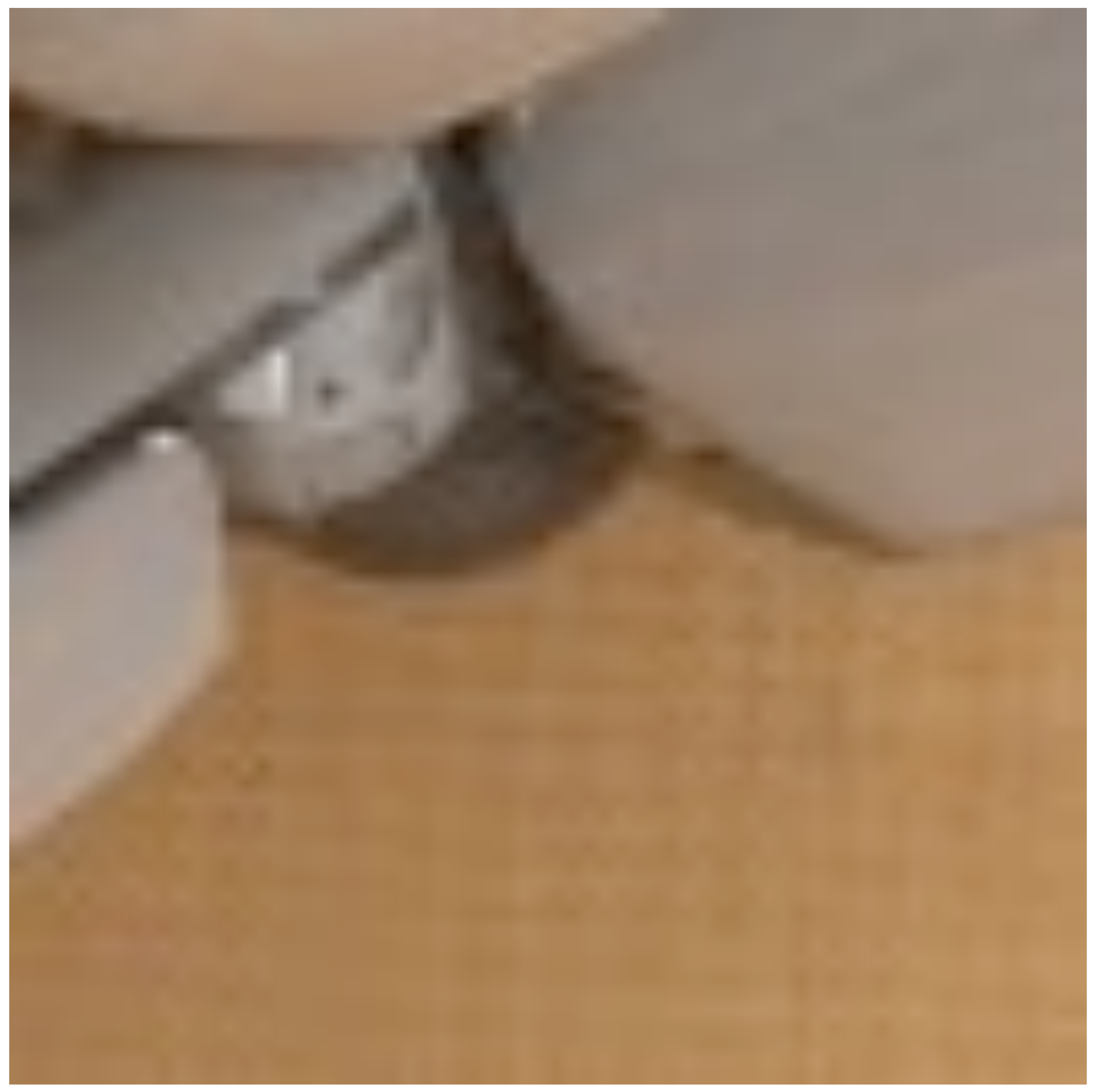
<SCRATCH???>

Limitations

Sliver re-rendering for interpolation

- Silhouettes
- Occlusions


Without re-rendering


Re-rendering occlusions

For interpolation one typically averages contributions from the two neighboring keyframes.
But at occlusions, only a single keyframe contributes and if the illumination changed strongly between the keyframes, a seam at the interface between occluded and un-occluded regions will appear.
<click>

<SCRATCH???>