Hello, everyone! I'm Shohei MORI at Graz University of Technology, Austria, and Keio University, Japan. I'm going to introduce the concept and practical aspects of Diminished Reality.

What is Diminished Reality?

The concept and why it matters

Diminished Reality (DR)

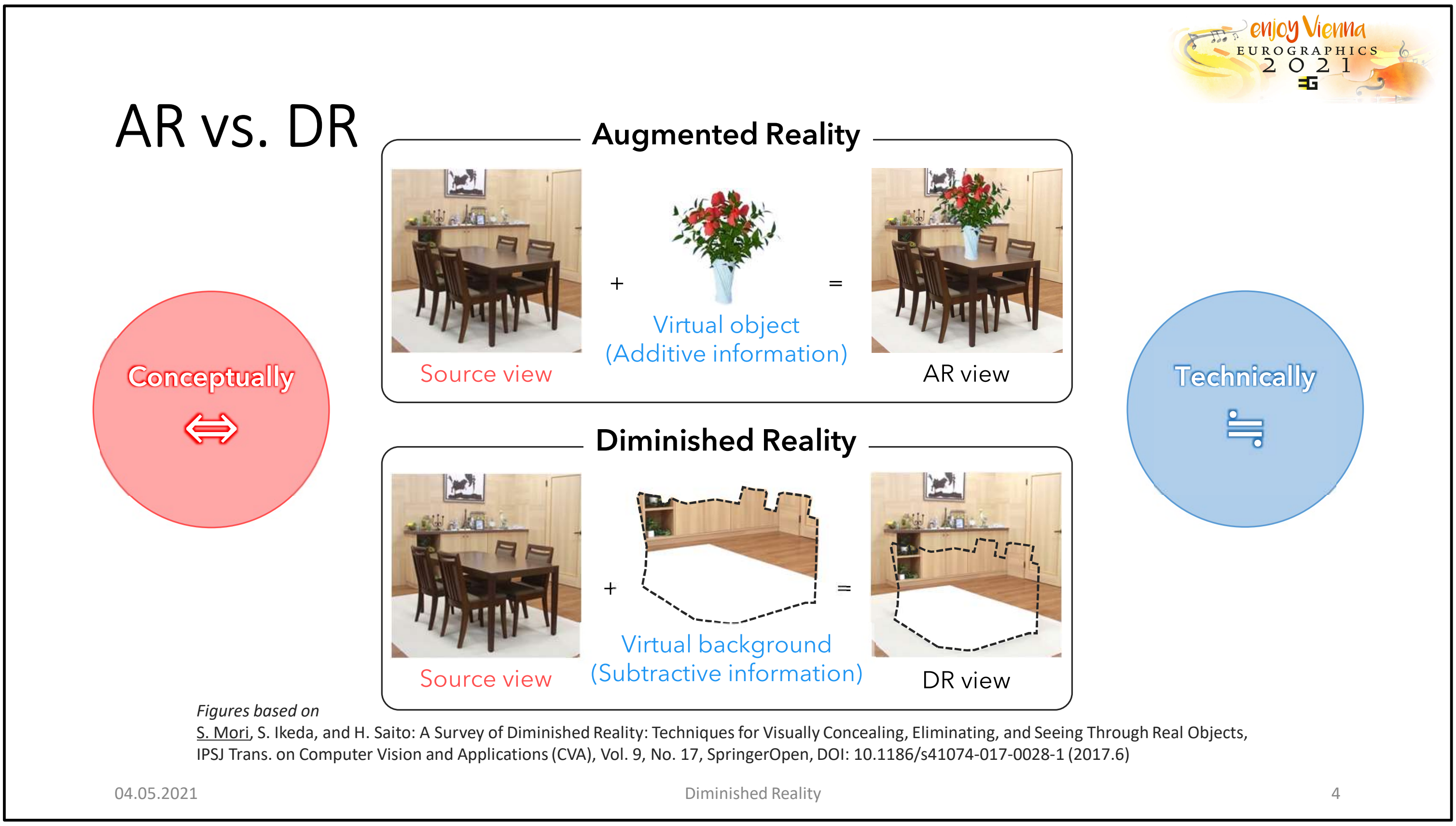
*"While most applications of AR are concerned with the addition of virtual objects to a real scene, **diminished reality describes the conceptual opposite** — namely, the seamless removal of real objects from a real scene."*

D. Schmalstieg and T. Hollerer (2016) Augmented Reality: Principles and Practice, Addison-Wesley Professional

DR is a set of methodologies for diminishing the reality, and concealing, eliminating, and seeing through objects in a perceived environment in real time.

S. Mori, S. Ikeda, and H. Saito: A Survey of Diminished Reality: Techniques for Visually Concealing, Eliminating, and Seeing Through Real Objects, IPSJ Trans. on Computer Vision and Applications (CVA), Vol. 9, No. 17, SpringerOpen, DOI: 10.1186/s41074-017-0028-1 (2017.6)

Let me first explain what Diminished Reality is. There are two lines from two pieces of literature. The book "Augmented Reality: Principles and Practice" reads "While most applications of AR are concerned with addition of virtual objects to a real scene, diminished reality describes the conceptual opposite – namely, the seamless removal of real objects from a real scene." The diminished reality survey paper defines DR in a bit broader sense, saying "DR is a set of methodologies for diminishing the reality, and concealing, eliminating, and seeing through objects in a perceived environment in real time." Steve Mann who introduced the concept and the name of Diminished Reality, first explained a diminished reality application with that we could diminish the colors of unimportant things so that we could focus on the remaining color part in our vision.

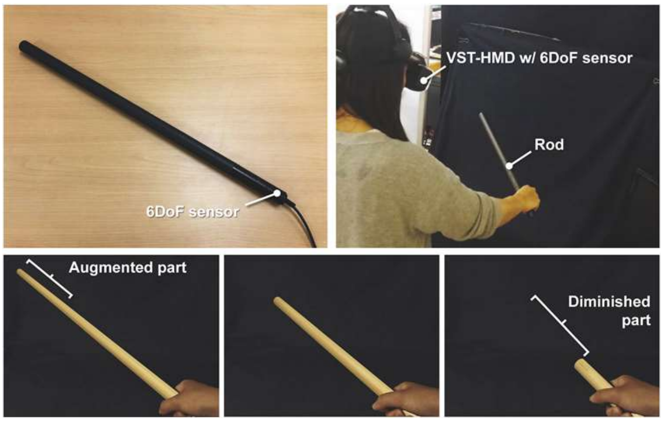


Considering the recent trends in DR as a technique to remove real objects, I would like to compare AR and DR. As we've learned so far in this tutorial, AR presents an augmented view. In this case, flowers with a vase are added to the scene. Instead, DR removes things from the real scene. In this example, the table and the chairs are removed. But to this end, what DR actually does is to add a virtual background image on top of the real scene, which is quite the same what AR does. This explains that AR and DR are the conceptual opposite as the AR book explains, while DR requires very similar technologies to those used in AR since DR also "adds" virtual information to the real scene. Here, what is added makes the conceptual difference.

Real-time Capability Matters!

A DR system must present an “experience” through multi-modal displays

- Usually targeting to **30Hz** refresh rate at **640 × 480** pixel resolution

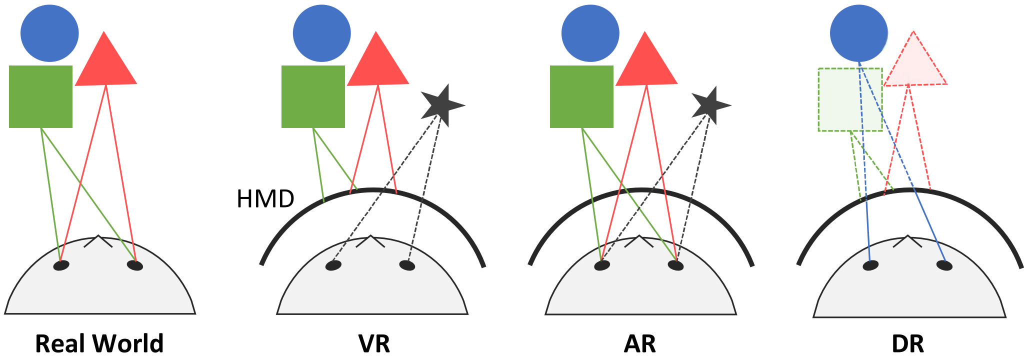


Figures based on
S. Hashiguchi, S. Mori, M. Tanaka, F. Shibata, and A. Kimura, “Perceived Weight of a Rod under Augmented and Diminished Reality Visual Effects”, Proc. The ACM Symp. on Virtual Reality Software and Technology (VRST) (2018.11)

Same as in AR, a DR system must present real-time augmentation. Here, “real-time” usually means to present images at 30Hz and the VGA resolution. Here’s an interesting example why going above the borderline gives us benefits. In this research, we virtually extended and shortened the stick and asked the participant to wave it and report how she felt. The results showed that the participants tend to feel the extended stick lighter than the original one and shortened one heavier. To present such an “experience”, the system throughput must guarantee the real-time operation. By satisfying real-time operations, definitely, DR can change our experience!

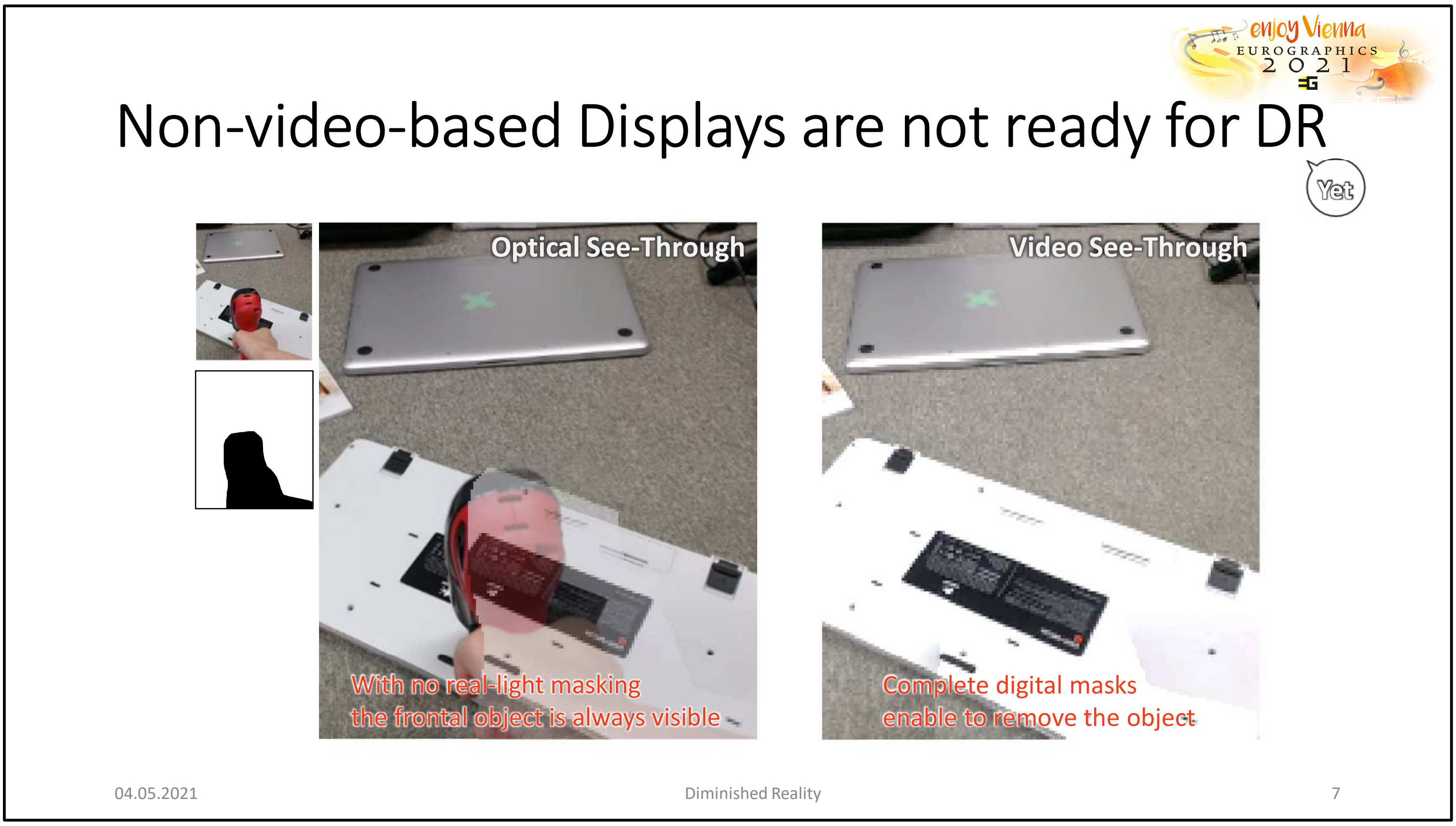
Displays for DR

- DR displays are capable of selectively occluding real light rays
- Light rays occluded by frontal objects need to be recovered virtually



Figures based on S. Mori and H. Saito, "An Overview of Augmented Visualization: Observing the Real World as Desired" APSIPA Trans. on Signal and Information Processing, Vol. 7, pages E12 (2018.10)

DR experiences are presented through a display. What makes DR unique here is that DR displays must selectively occlude real light rays and instead recover and present light rays originally occluded by the removed objects. As you might notice, AR and DR can be mixed, for example, to replace a LEGO character with the virtual counterpart to animate the character virtually through the display.



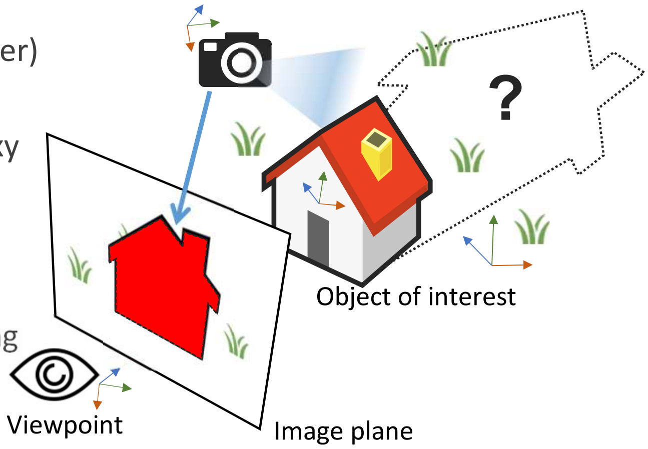
You will soon learn differences between the optical see-through and video see-through displays in detail in the later session of this tutorial, but here I phrase that the optical see-through combines a real scene and virtual imagery through optical combiners such as a half mirror, and the video see-through mixes real and virtual imagery as a video processing of video streams from the attached cameras. Although DR displays must block real light, optical combiners inevitably pass through environmental light. As such, I would note that optical see-through displays do not suit DR. In fact, the DR survey paper reports that almost all DR systems are implemented with video see-through displays.

How to Implement a DR System

Where to start and how to complete

Implementing a DR System

1. Tracking
   - Camera or scene tracking (e.g., vSLAM / marker)
2. Background proxy modeling
   - Planar proxy / multi-plane proxy / full 3D proxy
3. ROI detection
   - User annotation / semantic-segmentation
4. Background synthesis
   - Image-based rendering / Homography warping
5. Composition
   - Intensity interpolation / seamless cloning / smooth alpha masking / lighting estimation



Figures based on
S. Mori and H. Saito, "An Overview of Augmented Visualization: Observing the Real World as Desired" APSIPA Trans. on Signal and Information Processing, Vol. 7, pages E12 (2018.10)

In order to implement a DR system, we need these five components. First, the camera or the scene needs to be tracked to define the geometry relationships between the scene elements. When there's an additional camera to peep the background, it will be registered to the system as well. To synthesize the background information, we need to describe the background, whose data structure depends on the background synthesis methods. Usually, the region of interest is defined by the users by letting them annotate on the screen on demand, or otherwise, automatic segmentation methods are used if the category of the object of interest can be pre-determined. Once the background data is extracted, a synthetic view is generated and overlaid within the region of interest to hide the frontal object. Finally, for removing the seams between the real and virtual regions, optical compositions are made.

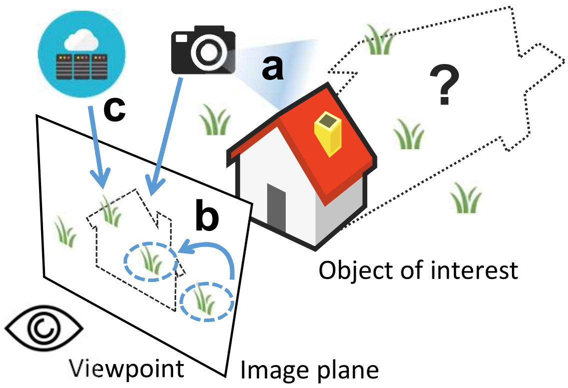
Background Resources

a) Multi-viewpoint images
- (+) Resources from *observations*
- (−) Hardware sync., calibration, color compensation, etc.

b) Pixels within the FoV (Inpainting)
- (+) No additional hardware, thus, portable
- (−) *Hallucinated* background
- (−) Fast (multi-view) inpainting is hard

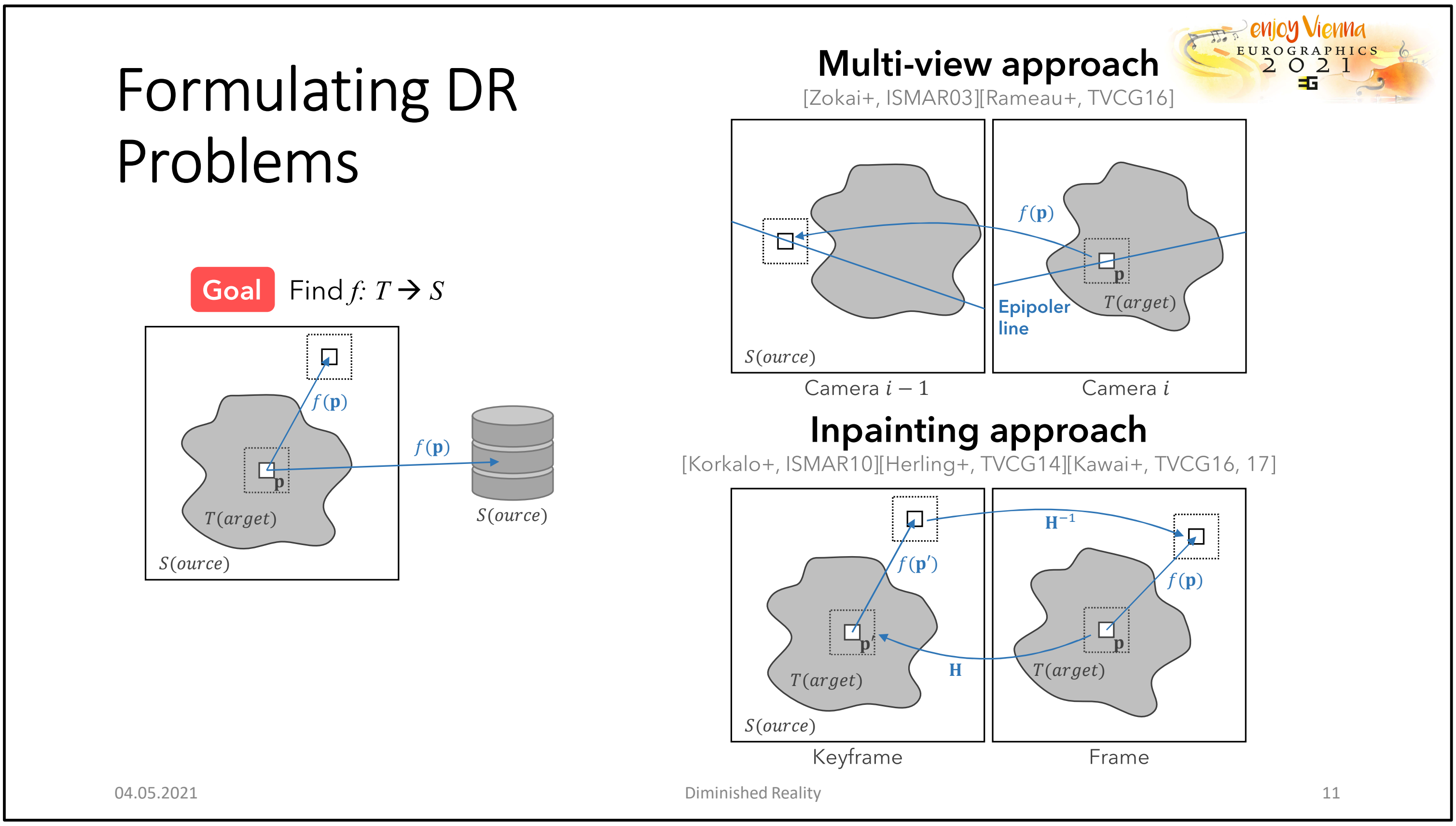
c) Dataset (Photo collection / Features)
- (+) On-demand resource
- (+) Well-prepared resources
- (−) Large memory or network connection
- (−) Day/time compensation

d) Combinations of the above



Figures based on
S. Mori and H. Saito, "An Overview of Augmented Visualization: Observing the Real World as Desired" APSIPA Trans. on Signal and Information Processing, Vol. 7, pages E12 (2018.10)

One of the biggest design choices to be made in DR is the selection of the background resources. Earlier works, like in the '90s, mainly used multi-viewpoint images to lend the benefits from 3D vision and image-based rendering techniques. Those approaches use additional cameras designedly placed in the environment or assume access to surveillance cameras that already exist in the environment. Those approaches can present real-time "observed" background changes while suffer from known issues in 3D vision and image-based rendering. Using image-inpainting, we can implement DR systems without preparing such additional cameras by sampling pixels from the currently captured image itself. However, we must extend the image-space processing for video streams captured under 6DOF camera motion for DR. If a dataset is available, we could access it to retrieve images of the same scene but taken on different days and times. Of course, we may combine those three approaches to have different benefits from different approaches.



So, DR can be defined as a problem of how to synthesize the background within the region of interest from available resources, which can be generalized like this. We have a target region T as the region of interest. Then, the pixel p within T refers to a pixel in the source region S. Such relation is described as f. The reference is not restricted to pixels within the field of view, but we could use an image dataset as I mentioned in the previous slide.

In multi-view approaches, calibrated cameras allow p to refer to pixels along the Epipolar lines in the resource camera frames for a fast pixel search.

In inpainting approaches, pixels p' refers to pixels within the field of view. If the calculation of f is fast enough, the reference f can be updated frame by frame, but this approach may lose the temporal coherence. To maintain the temporal coherence, recent approaches use a keyframe where the initial inpainting is done, and for the next frames, they warp the keyframe to use it as a reference frame in the current frame. How to use the keyframe is another design choice with a trade-off between speed and quality.

Since the latest work in DR rather focuses on inpainting-based approaches, the rest of my talk will be about the inpainting approaches and the design choices to balance the speed-quality trade-off.

Inpainting-based DR

Inpainting scene objects in real time

Fast Inpainting for Marker Hiding

S. Siltanen, “Texture Generation over the Marker Area”, Proc. ISMAR, 2006.

Try this out:
https://github.com/Mugichoko445/DRMarkerHiding

This video demonstrates one of the earliest works of marker hiding application using a simple and fast inpainting algorithm. Note that conventional image inpainting algorithms usually take seconds to minutes due to exhaustive searching or semi-optimal nearest neighbor searching. With the marker hiding application, we could present a virtually markerless AR in 6DOF. Here, I overlay 3D axes of the marker origin to show where the marker is placed. The right video shows a predefined image space where the actual inpainting is happening in real-time. This is my implementation of the approach by Siltanen presented at ISMAR 2006 and will be available online in the link.

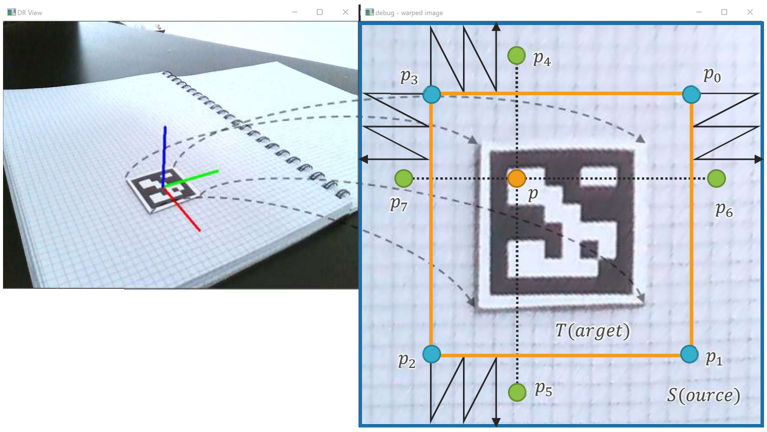
This video shows the high interactivity and some limitations such as a lack of temporal coherency, ghosting, and distorted textures on the marker area.

Fast Inpainting for Marker Hiding

S. Siltanen, “Texture Generation over the Marker Area”, Proc. ISMAR, 2006.

A pioneering marker hiding method

Mirroring and mixing the vicinity pixels towards the marker region



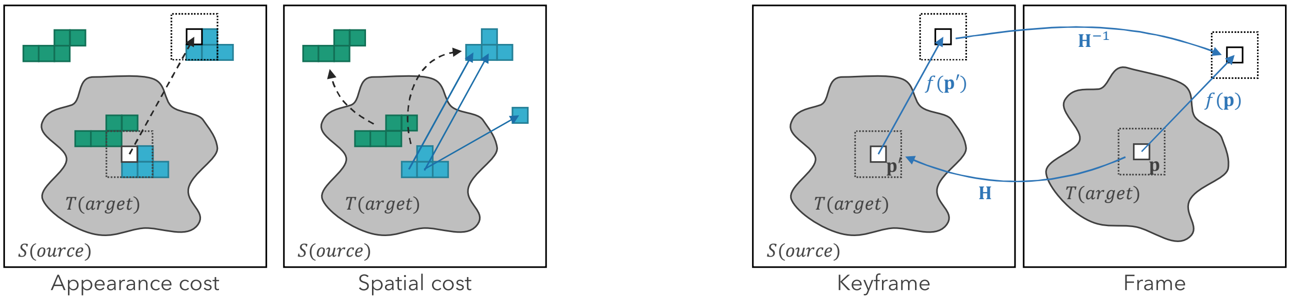
$$p = \sum_{i=\{0,\ldots,7\}} w_i p_i$$

This approach first transforms the input image on the left to a pre-defined image space on the right using Homography warping to remove the perspective distortion. The marker area to be filled with pixels in the vicinity area is a slightly larger rectangle region than the original marker region surrounded by p0 to p3 in blue. The pixel p is then a weighted sum of the eight surrounding pixels from p0 to p7. Starting from the left top corner of p3, p raster-scans the target region and vicinity pixels p4 to p7 moves accordingly within the source region in zigzag lines. This operation of mirroring pixels within the source region is considered reference function f. After filling in the target region, the image is warped back to cover the marker area. Clearly, the textures need to be aligned with the zigzag raster scans in the source region. Otherwise, artifacts as shown in the video result will appear.

PixMix – A Keyframe-based Approach

J. Herling and W. Broll, "High-Quality Real-Time Video Inpainting with PixMix," IEEE TVCG, Vol. 20, Issue 6, pp. 866 - 879, 2014.

- Inpaint a frame (Keyframe) and **warp** it to the current frame as a reference
- Keep copying adjacent pixels when good pixels are not found
- Region-wise parallel pixel updates in an image



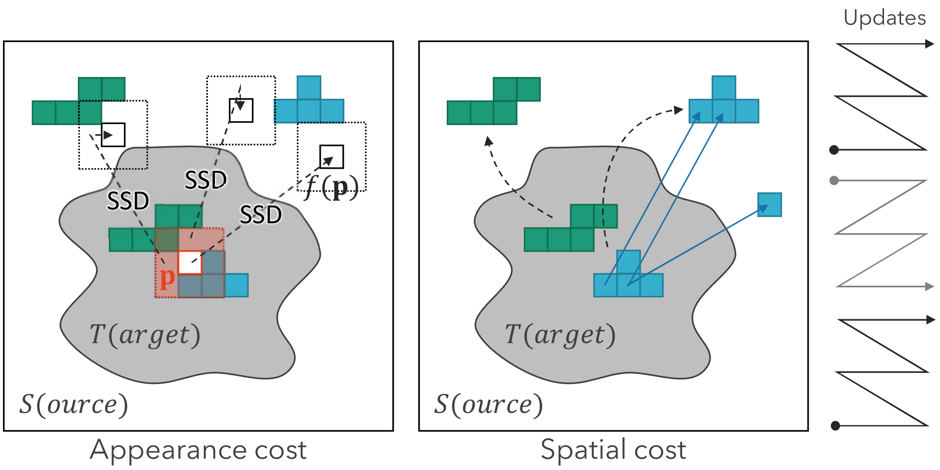
$$\min_f \sum_{p \in T} cost_{\alpha}(p) = \alpha\, cost_{appearance}(p) + (\alpha - 1)\, cost_{spatial}(p)$$

04.05.2021 Diminished Reality 15

Dissimilarity-based pixel nearest neighbor search algorithms would potentially solve the problem because collected pixels do not have to be aligned in a predetermined order like in Siltanen's approach. PixMix takes that direction and it is a similar approach to a famous inpainting algorithm called PatchMatch but significantly speeds up the entire operation with some assumptions. PixMix optimizes f as to collect best describing pixels in terms of the texture appearance, meaning that it tries to match the patch in the source region to that in the target region. In addition, it defines a spatial cost term that tries to connect adjacent pixels so that it copies the next pixels by giving up further pixel searching that would end up with worse candidates. PixMix processes the first frame and warp the frame to the current frame to use it as the initial guess to inpaint the upcoming frames. The use of a keyframe can prevent from breaking the entire texture structure while it can update the pixel colors with those in the current frame.

PixMix – Optimization

J. Herling and W. Broll, "High-Quality Real-Time Video Inpainting with PixMix," IEEE TVCG, Vol. 20, Issue 6, pp. 866 - 879, 2014.



For more details: https://github.com/Mugichoko445/PixMix-Inpainting

Because the latest DR approach that I will introduce you later also uses similar optimization step, I would like to dig into the algorithm a bit more. First, f, which describes per pixel locations from T to S, is randomized and we want to update all pixels in T by scanning from the top left to the bottom right. Let's say, a pixel p now refers to here with f. We define the current patch at p and the horizontal and vertical candidates, which may have a better estimate as they have been updated before we reach the current p. We compare the current patch and the other patches located at jumped locations by referring to the current f by sum of squared differences.

As the spatial cost, distances between referred pixels are calculated. The two blue pixels that keep the spatial relationship have the minimum cost and if the relation breaks, those pixels are penalized by the cost. That is, pixel locations must be kept to keep the spatial cost low.

On each pixel, we take the pixel location that takes the minimum cost. Minimizing the spatial cost could guarantee at least a good connectivity of pixels, since we still can use next pixels as a fall back solution. By this, we could omit further pixel searching by more random samples as done in the conventional PatchMatch algorithm. Therefore, we could expect fast convergence for the optimization.

This process is done in bidirectional raster scans, and using a image pyramid. Also, pixels are updated in separate regions within the field of view in parallel to further speed up the pixel searching process.

PixMix – Optimization

The original pictures by Harry Strauss (https://pixabay.com/ja/users/image4you-2459255/) from Pixabay

This video demonstrates the optimization steps in an image pyramid. Given an input image and a mask image, PixMix optimizes the pixels within the region of interest. According to the paper, this process completes in several tens of milliseconds on average. Note that the image pyramid is built only once at the keyframe. In the next frames, we could use the guess from the keyframe to directly optimize the original-sized image.

Marker Hiding Using PixMix

Try this out:
https://github.com/Mugichoko445/DRMarkerHiding

My implementation of the PixMix algorithm runs on the marker hiding application. The left video shows the DR view and the right video shows the warped keyframe. Here, I sometimes restart the algorithm to get new results. Although, in the warped keyframe, pixels are baked and just warped to fit the appearance of the current frame, PixMix updates those pixels to new ones within the current frame by referring to updated pixel locations based on the transformation map f. It seems that the inpainted pixels are jiggling because of the inaccuracy of the marker. New pixel locations are located apart from the marker, and therefore, if the marker detection is erratic, such locations will jump from locations to locations. Overall performance looks OK. Compared to the Siltanen's approach, the inpainted textures look clearer.

Multi-plane Inpainting

N. Kawai, T. Sato, and N. Yokoya. "Diminished Reality based on Image Inpainting Considering Background Geometry”, IEEE TVCG, Vol. 22 Issue 3, pp. 1236 - 1247, 2016.

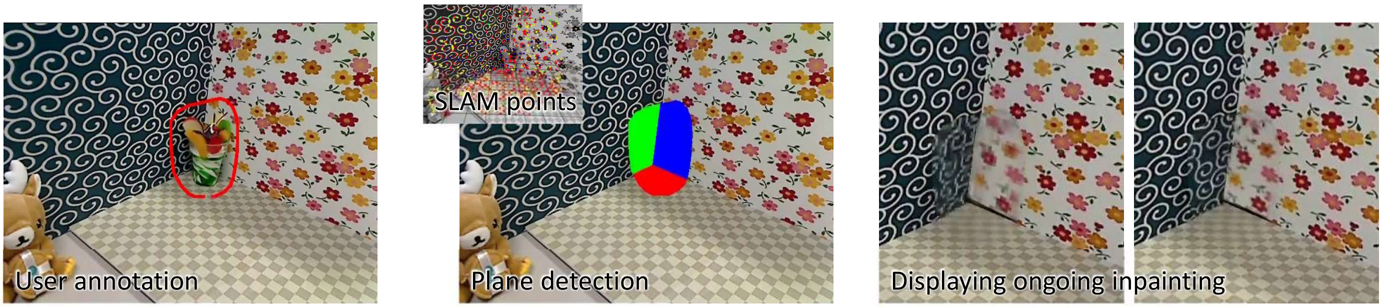
Video: Courtesy of Dr. N. Kawai

Apart from the inpainting algorithm itself, we should take into account for more complex geometry proxies than a plane to extend the application range of our DR system. One idea is to approximate scene geometry to several planes so that we can run an image-inpainting algorithm in each plane.

Multi-plane Inpainting

N. Kawai, T. Sato, and N. Yokoya. "Diminished Reality based on Image Inpainting Considering Background Geometry", IEEE TVCG, Vol. 22 Issue 3, pp. 1236 - 1247, 2016.

- Inpaint the ROI on **independent plains** in a **keyframe**
- Tracking & inpainting on different **threads**
- Show **intermediate** inpainting results



Figures: Courtesy of N. Kawai

Kawai et al. proposed to detect scene planes from a SLAM point cloud by analyzing the normal vectors and run inpainting on each plane so that pixels from different planes will not mixed. Because their inpainting method takes several seconds to complete like the other conventional image-inpainting approaches, they distribute the inpainting process in a different thread running in the background. An interesting visual effect they implemented is that, while the system processes an image pyramid, the system reveals the ongoing results on the DR display so that the user does not really have to wait till the inpainting completes.

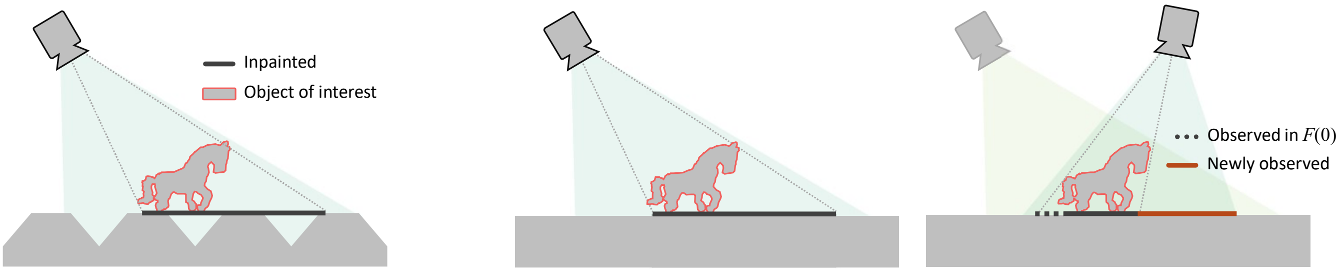
Marker Hiding Using Multi-threading

Try this out:
https://github.com/Mugichoko445/DRMarkerHiding

In this marker hiding, I implemented the visualization technique using multi-threading with that the system individually executes the marker detection and inpainting in a rectified keyframe using a variant of PixMix. Upon completion of inpainting at each pyramid level, the inpainting thread sends back the intermediate inpainted image to the main thread, so that the main thread can display the current result immediately in the DR view. The main thread also runs the Poisson seamless blending to resolve the color inconsistency due to the viewpoint changes. In this implementation, I added a random sampling step during the PixMix optimization to find more plausible pixels, but it slows down the entire process. This motivates the multi-threading technique by Kawai et al. to be introduced.

Plane(s) as Background Geometry Proxy?

- Image-inpainting works in an image-space
- Limitations to AR/DR
  - No interaction with the background after a DR method is applied
  - No automatic updates when new real object pixels are observed
- How can we extend inpainting for 3D AR scenes?



Keyframe inpainting | First keyframe $F(0)$ | Upcoming frames

While the inpainting-based DR methods so far successfully remove objects, there are some clear limitations. The most obvious limitation comes from the fact that geometry is approximated to a plane or at the best several planes. The main reason why inpainting-based DR only runs in such scenes is that the image-inpainting is designed for an image-space processing. Due to this constraint, no interaction with the background such as relighting and physics animations in DR has not been made. Further, inpainting-based DR methods often assume a flat object to be removed. Therefore, newly observed areas remain inpainted while the real geometry colors have been revealed already.

InpaintFusion – 3D Inpainting for AR Scenes

S. Mori, O. Erat, W. Broll, H. Saito, D. Schmalstieg, and D. Kalkofen, “InpaintFusion: Incremental RGB-D Inpainting for 3D Scenes”, IEEE TVCG, Vol. 26, Issue 10, 2020.

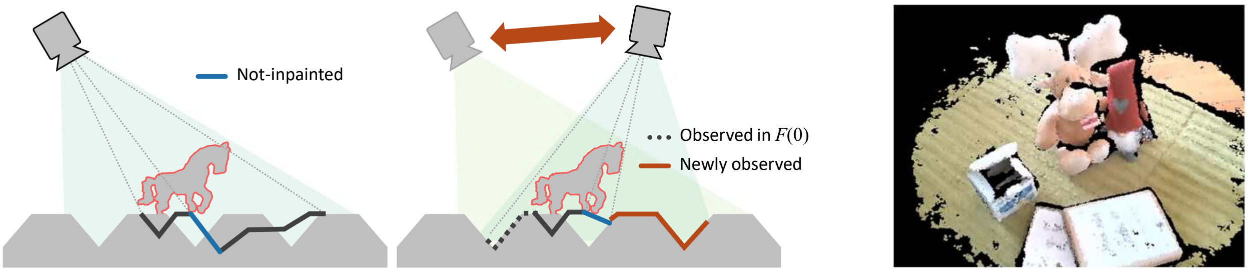
One solution from the latest approach to solve these issues is to combine screen-space depth inpainting and depth fusion to explicitly describe the scene depth even in the hallucinated area. This video result of the InpaintFusion approach shows color and the depth space inpainting in real-time. Notice that the headlight of the virtual car lit the inpainted area and the snowballs interact seamlessly with the real and the hallucinated background. In the original video streaming, the toy horse still exists there.

Ideas

S. Mori, O. Erat, W. Broll, H. Saito, D. Schmalstieg, and D. Kalkofen, “InpaintFusion: Incremental RGB-D Inpainting for 3D Scenes”, IEEE TVCG, Vol. 26, Issue 10, 2020.

Multi-keyframe inpainting with **RGBD fusion** and an **IBR** technique
Image-Based Rendering

- *RGBD inpainting* per keyframe
- Filling in missing pixels in the *next keyframes* and *fuse* them
- Pixel *blending* based on view-/surfel-priorities



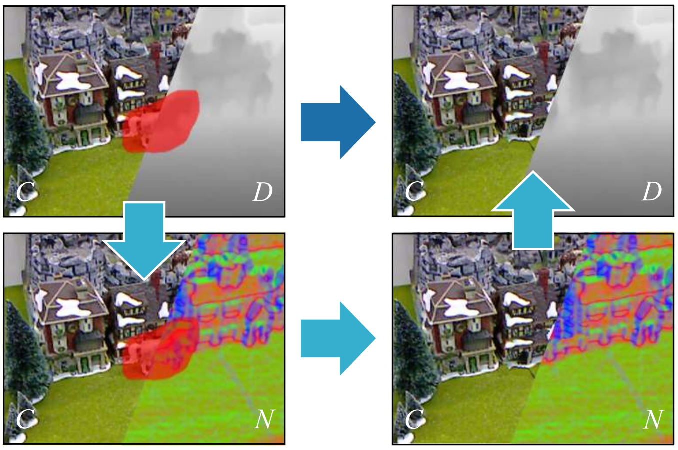
First keyframe $F(0)$ | Upcoming frames | Surfel fusion

Here are several key ideas to achieve 3D inpainting for AR. First, the inpainting is done both in color and depth space to obtain interactable inpainted areas. With single keyframe inpainting, we might have missing pixels when we move the camera. Therefore, additional keyframes are inserted. Now the question is how to preserve the consistency of the keyframes. InpaintFusion uses a fusion approach to integrate the inpainted depth values as a global scene map. Finally, obtained inpainted keyframes are blended using an image-based rendering technique, which is similar to view-dependent texture mapping but with the highest priority to the actually observed surface colors if available.
During the run-time, the DR system uses the multi-threading technique to run the RGBD inpainting in a different thread and runs only the image-based rendering and tracking as the main processing. Therefore, the system runs in real time.

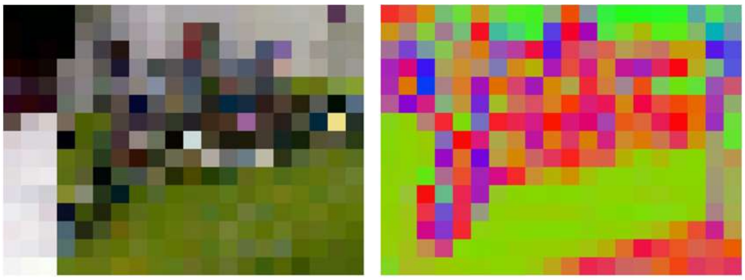
RGBD Keyframe Inpainting

S. Mori, O. Erat, W. Broll, H. Saito, D. Schmalstieg, and D. Kalkofen, “InpaintFusion: Incremental RGB-D Inpainting for 3D Scenes”, IEEE TVCG, Vol. 26, Issue 10, 2020.

- RGBD inpainting via RGB-Normal inpainting
- Depth from depth gradient samples from f^*



$$f^* = \arg\min_f \sum_{\mathbf{u}\in T} \left(\underbrace{w\rho_t(f,\mathbf{u})}_{\text{Texture term}} \underbrace{\rho_g(f,\mathbf{u})}_{\text{Geometry term}} + (1-w) \underbrace{\rho_s(f,\mathbf{u})}_{\text{Spatial term}} \right)$$



The goal of RGBD inpainting is, needless to say, to obtain inpainted color and depth values within the target region. To avoid direct sampling of depth values captured under perspective distortions, we need to inpaint normal values and then recover the depth from sampled depth gradients. In other words, inpainting is done in color and normal spaces first. The optimization step to find a best pixel reference map f is similar to the PixMix approach. The minimization includes texture term and spatial term as in PixMix algorithm, but now a geometry term is introduced to optimize the normal values. The geometry term here is simply an SSD of sampled patches of normals. After filling in colors and normals, depth values are recovered by solving the Poisson equation using depth gradients sampled via the optimized f.

Keyframe (KF) Propagation and Blending

S. Mori, O. Erat, W. Broll, H. Saito, D. Schmalstieg, and D. Kalkofen, “InpaintFusion: Incremental RGB-D Inpainting for 3D Scenes”, IEEE TVCG, Vol. 26, Issue 10, 2020.

- KF is inserted when the sensor gets away from the closest KF
- KF’s transformation map f is transformed to a new KF
- Multiple KFs are **blended** over the inpainted global surfel map

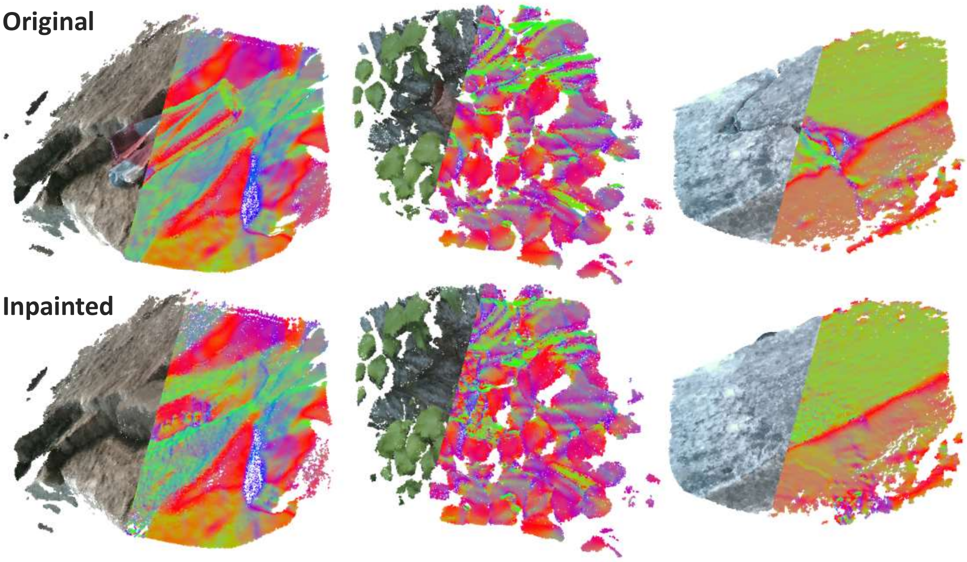
After the initial keyframe inpainting, a new keyframe is inserted when the sensor gets away from the keyframe. In the video example, black non-inpainted area reveals while moving the sensor. Then, a new keyframe inpainting completes the missing pixels.
In this step, the first keyframe is transformed to the second keyframe to initiate the new inpainting process. For a plane geometry proxy, we can use Homography warping. Here, instead, we use a latest global map from the SLAM system as the 3D geometry proxy.
This video shows three inpainting results of the InpaintFusion approaches. The multiple depth frames are fused as a global map. The color frames are blended using an image-based rendering approach, as I mentioned before. Therefore, the temporal coherency is kept over the frames.

Fusion Results

S. Mori, O. Erat, W. Broll, H. Saito, D. Schmalstieg, and D. Kalkofen, "InpaintFusion: Incremental RGB-D Inpainting for 3D Scenes", IEEE TVCG, Vol. 26, Issue 10, 2020.

Original

Inpainted



For **tracking** and **user-annotation** (ROI)

For **3D AR interactions**

04.05.2021 Diminished Reality 27

The system preserves the original 3D reconstruction to track the scene without any disturbance from the inpainted pixels. This is done by adding label information to each surfel to distinguish surfels from sensors and surfels from the inpainting process.

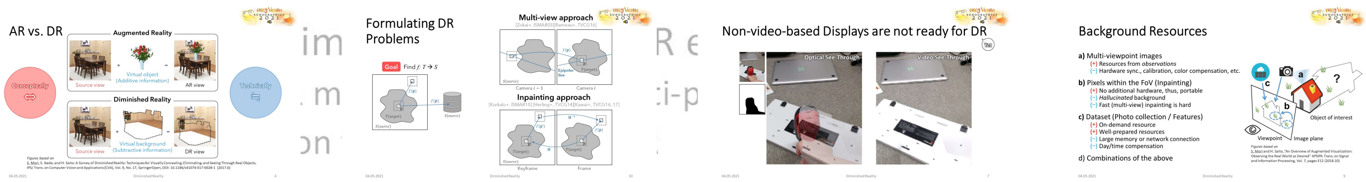
Different from the previous plane-based approaches, this system can remove 3D objects in 3D scenes. The generated surfels from the inpainting process have another label. Therefore, those surfels with labels of inpainting and sufels with none user-annotated labels are combined to generate interactable surfeces for AR applications. The bottom images show such inpainted surfeces.

Summary

Reviewing and take-home messages

Summary

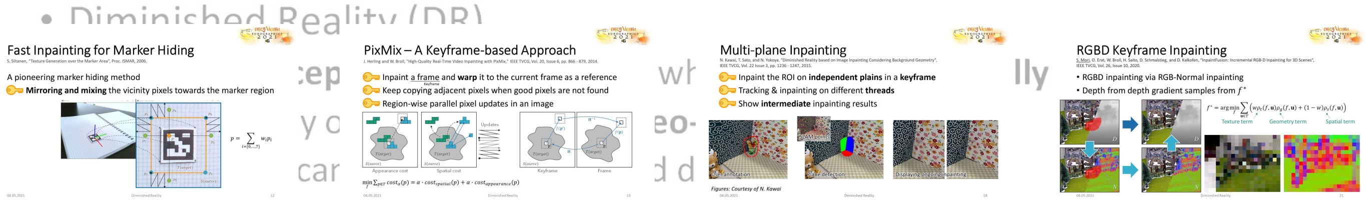
- Diminished Reality (DR)
  - DR is a **conceptual opposite** to AR, while they are **technically similar**
  - The majority of DR systems are **video-based**
  - Multi-view cameras, inpainting, and dataset



Summarizing my talk, I introduced you the concept and the technical aspects of Diminished Reality by glancing different approaches from papers. Compared to AR, DR is a conceptual opposite, since DR removes real objects from a scene in real-time. However, to build a DR system, the system consists of similar components to AR systems. That includes scene tracking and composition of virtual imagery to the real scene. The technical major differences may include that the DR systems additionally require finding an optimal transformation map f describing pixel relationships between the source and the target regions.

Since real-light blocking in AR or DR displays is still an unsolved problem in practice. Therefore, we need to implement our DR systems using video see-through displays. I introduced various background image resources, but in this lecture, I focused on inpainting approaches as it's the latest trend in DR. For surveying other DR research, I would recommend to look into the DR survey paper from CVA journal.

Summary



- (Semi-)Real-time inpainting for DR experiences
  - Mirroring & mixing, keyframe, multi-plane approaches
  - InpaintFusion for full 3D DR and AR

I introduced four approaches, starting from a very simple and fast algorithm by Siltanen. PixMix by Herling and Broll achieves a very fast pixel searching which is less restrictive in texture patterns in a scene. Kawai et al. extended an image-space inpainting approach for multi-plane scenes. Finally, InpaintFusion achieves full 3D inpainting using RGBD inpainting, a fusion technique, and image-based rendering to generate interactable 3D surfaces after the inpainting. As a short note to compensate for missing parts in this lecture, please notice that I have skipped talking about object annotation techniques to define a region of interest and color compensations often used for more seamless composition.

Take-home Message

- DR is a missing piece that compensates AR
- Real-time 3D inpainting is still challenging
- All inpainting-based DR systems rely on exemplar-based approaches
- Multi-modal DR is an un-touched research area

Try out the marker hiding applications:
https://github.com/Mugichoko445/DRMarkerHiding

With DR techniques, you can make AR more flexible and applicable. If you want to interact with a real object, let's say, if you want to push a cup on a table to clear a path for your virtual character, the system should firstly remove the cup using a DR technique and overlay a scan of the cup on a shifted location using an AR technique. Otherwise, the only way for your virtual character is to detour the real cup to go forward.

Yet, real-time 3D inpainting is challenging. As a fact, we are still running our algorithms on 640x480 pixel resolutions, which are far from the eye resolution. 30Hz frame update is also way slow considering our eye ability. Also, inpainting must run on-demand which makes it difficult to specify a domain to be applied.

You might have noticed that all inpainting-based DR uses exemplar-based inpainting approaches. One may explain that it's because of the flexibility and applicability of the exemplar-based approaches. AR or DR users will run inpainting on demand, and because of this, the context of inpainting may diversely changes. Tabletop, urban street, and maybe in a bathroom. Still, considering the recent progress in learning-based image synthesis such as neural rendering, we cannot stop expecting progress in DR as well in the near future.

This is a bit out of the topic at Eurographics, but as I suggested at the beginning, DR experiences may have some effects on our perception. DR may change our experience and therefore how we see the world. Apart from changing visuals of a real environment, we can consider changing the other modalities such as sounds and haptics. Those aspects have not been investigated deeply yet.

For people who feels difficulties in coding the DR applications, I will prepare example codes here in the GitHub. It will be a good starting point for students and beginners in DR. Thank you!