Scan&Paint: Image-based Projection Painting

Vanessa Klein* Markus Leuschner Tobias Langen Philipp Kurth Marc Stamminger Frank Bauer†

Friedrich-Alexander University Erlangen-Nürnberg (FAU), Visual Computing

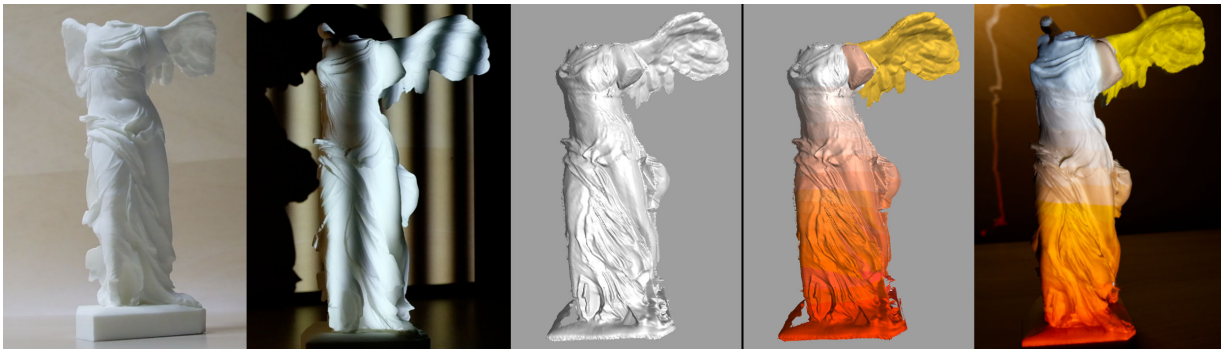


Figure 1. Our pop-up system iteratively scans white target objects and continuously fuses the reconstructed depths in screen space for novel synthetic views by applying the FragmentFusion algorithm. The user's digital paint is incorporated into the original scan views and included in the fusion process. The projector projects the color onto the tracked real-world object and luminance-adjusts the color to the estimated surface geometry.

Abstract

We present a pop-up projection painting system that projects onto an unknown three-dimensional surface, while the user creates the projection content on the fly. The digital paint is projected immediately and follows the object if it is moved. If unexplored surface areas are thereby exposed, an automated trigger system issues new depth recordings that expand and refine the surface estimate. By intertwining scanning and projection painting we scan the exposed surface at the appropriate time and only if needed. Like image-based rendering, multiple automatically recorded depth maps are fused in screen space to synthesize novel views of the object, making projection poses independent from the scan positions. Since the user's digital paint is also stored in images, we eliminate the need to reconstruct and parametrize a single full mesh, which makes geometry and color updates simple and fast.

Index Terms: Computing methodologies—Mixed / augmented reality; Computing methodologies—Image-based rendering; Computing methodologies—Reconstruction

1 Introduction

Rapid prototyping is essential in collaborative shared environments to save time and costs. In this regard, dynamic projection mapping offers real-world and haptic interactions by projecting user-generated content on a target object. For example, instead of ordering several different prototypes of manufactured products, e.g., shoes, projection mapping enables designers to visualize and discuss different colors and materials on a single manufactured prototype. However, while it is easy to set up spatial augmented reality (SAR) for simple (e.g., planar) or at least static target objects, projection mapping for rigid, but movable and arbitrarily shaped, three-dimensional objects currently requires more involved preparations. A pixel-correct 3D model of the target object and calibrated object tracking are required to correctly compensate for varying irradiance at uneven surfaces and to avoid artifacts along inner silhouettes. Additionally, painting or texturing the objects needs further pre-processing like generating suitable UV maps [10].

Knowing the target surface, scanning and preparing it in advance is, at best, a tedious and time-consuming preliminary work that demands trained users. It is also tempting to save preparation time by reconstructing only the distinct parts of the surface that are presumably needed for projection – only to later discover that a new spontaneous idea would require more reconstructed surface. With current systems it is thus not feasible to start unscheduled projection mapping sessions with unknown objects, for example, designers meeting spontaneously to quickly discuss new design ideas on a just delivered prototype. It also prohibits mobile setups, with which objects and surfaces can be annotated on the spot.

Instead of having a projection mapping system require a high-quality full scan, the system itself should decide when and which parts of the surface require reconstruction to be fully flexible for any kind of user application. We therefore design a pop-up projection mapping system that does not rely on prepared scans, but serves as an automatic on-demand 3D scanner. This enables us to iteratively scan a movable object and paint onto it during runtime. Our method follows the zeitgeist of other more traditional Augmented Reality approaches where users prefer to augment the virtual world immediately, without a lengthy preparation phase at the beginning.

Instead of reconstructing a single complete model from the recorded scans, we leverage depth and color fusion – similar to image-based rendering – to compose the projection content. The only remaining requirement is a geometrically calibrated projection mapping system (a projector and a tracking/scanning device), most conveniently realized by installing the components on a tripod or any other rigid mounting system.

We summarize our contributions as follows:

- The first of its kind fully functioning **system** that allows immediate, geometry-aware projection painting on an unknown, complex and movable rigid target object with consumer-grade hardware

*e-mail: vanessa.klein@fau.de
†e-mail: frank.bauer@fau.de

This is the accepted version of the following article: V. Klein, M. Leuschner, T. Langen, P. Kurth, M. Stamminger and F. Bauer, "Scan&Paint: Image-based Projection Painting," 2021 IEEE International Symposium on Mixed and Augmented Reality (ISMAR), 2021, pp. 517-525, doi: 10.1109/ISMAR52148.2021.00069.

- A **paradigm shift** that proposes image-based instead of mesh-based projection mapping
- A **trigger mechanism** that automatically issues discrete scene captures, if the current surface estimate does not suffice
- A **novel backward-oriented application** of the **FragmentFusion** algorithm [19] to store color data for 3D surfaces

2 Related Work

Projection mapping renders a digital scene and projects it onto a real-world surface. In contrast to, e.g., a planar wall, projecting onto arbitrarily shaped objects requires knowledge about this shape to counterbalance real-world shading effects and was first performed by Raskar et al. [17]. Later, Siegl et al. introduced geometry-aware projection mapping in screen space for multiple projectors and movable, known target objects [22]. Dynamic setups track their known objects with the help of markers [1] [14] [25] [24], by fitting their models into depth camera frames [22] [11] [28], by performing feature matching for feature-rich textures [18] or are optimized towards the object, e.g., face tracking [3]. The MIDAS-projection system by Miyashita et al. [12] projects onto unknown and even deformable surfaces like clay or fluids. In addition to uniform colors or materials, tileable textures with spatio-temporal consistency are projected while the target surface is trackable via optical flow. In contrast to their work, our system cannot handle deformable surfaces but performs persistent texturing, i.e., textures remain at the correct location after re-entering the projector frustum. Our incremental surface estimate also enables multi-view screen-space techniques like shadow mapping. Recent advances in the field are summarized by Grundhöfer et al. [4]. Projection painting, i.e., providing a user with means to color the projection during runtime, is demonstrated by Raskar et al. [2] and Lange et al. [10] [9], but requires a full, parametrized mesh. MadMapper and Lightform are, amongst others, tools available to the general public that combine visible structured light scanning, projection mapping and content design. However, they do not support real-time tracking and are thus limited to a single point of view and static objects.

Geometry-aware projection mapping requires high-quality surface data. 3D scanning, i.e., retrieving digital color and shape data from real-world objects, is a well studied, but at the same time up-to-date research topic. The best scanning method depends on the application itself and its precision requirements. Structured light scans represent a compromise between speed and quality, while at the same time requiring only consumer-grade hardware. Predefined patterns are projected onto the target object and are then recorded by a camera. Leveraging the knowledge about the patterns, the depth is reconstructed from the images. Gray code scanning provides good-quality results, but the number of pattern images depends on the projector's spatial resolution, which increases the total scan duration. Phase shifting algorithms [23], on the other hand, decouple the patterns from the projector resolution and are often faster. Zhang gives a detailed overview on structured light methods [27]. Chebyshev Phase Shifting by Moreno et al. [13] is a fast technique that requires nine patterns (three if color-encoded) and provides high-quality scans.

A single structured light scan is usually made from a fixed view direction. Thus, multiple scans from different perspectives are needed to scan an entire three-dimensional object. Due to small errors in each scan, a single dense reconstruction from multiple scans is no trivial task. Volumetric methods like KinectFusion [15] [5] fuse geometry into global structures, but generally exhibit a large memory footprint. Zollhöfer et al. [29] give a comprehensive summary of recent techniques in 3D reconstruction. FragmentFusion by Rückert et al. [19] does not attempt to reconstruct a global model, but fuses geometry (and color) in screen space by computing a weighted average

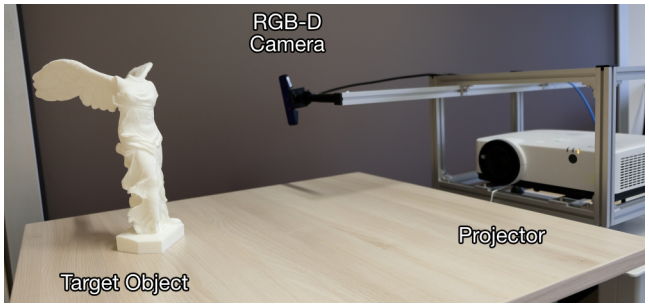


Figure 2. Projection mapping setup. RGB-D camera and projector face an arbitrary (white Lambertian) target object.

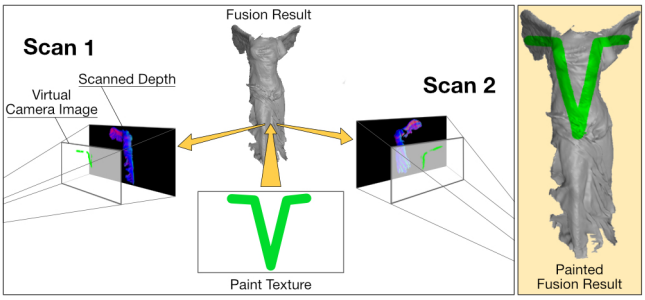


Figure 3. Overview. Scans from various perspectives are fused for new synthetic views (e.g., projection image). Paint drawn into the paint texture is backpropagated to the original scans (virtual camera images) and from then on included in the fusion process.

of fragments lying on the same surface. This approach reminds of image-based rendering techniques, summarized by Shum et al. [21].

We combine structured light scanning, fusion-based rendering and dynamic, geometry-aware multi-projection mapping to project onto unknown, complex and movable surfaces (see Fig. 1). In contrast to former projection mapping scanner systems [26] [16] [7] [6], our system is not based on a single, fully reconstructed mesh, but it scans the moving surface repeatedly during runtime and fuses continuously in screen space.

3 System Overview

Our projection mapping setup consists of an RGB-D camera and a projector (see Fig. 2). The target object is tracked via the depth channel and scanned by observing projected structured light patterns via the color channels. We calibrate camera and projector intrinsically and extrinsically [8].

Our system operates in an iterative manner: scan, paint, move, (scan,) paint, move, (scan,) paint, and so forth. On start-up, a first structured light scan of the target object is performed automatically (see Sect. 4). The reconstructed depth is shown to the user on an external display where they are also given the tools to digitally paint it. By rendering the scene from the projector's point of view, this first scan view is already suitable for projection mapping. However, a single structured light scan covers only the surface that is visible to both camera and projector. By moving the object, new parts of the surface are exposed to the camera-projector pair, triggering a new scan if necessary. The new reconstructed depth map yields a new perspective of the target object. Since we track the object via the camera's depth channel, the position of the previously recorded scan is known to the system and the depth maps are aligned to each other.

This iterative process is only realizable with a system design that allows to incrementally add newly scanned geometry. Furthermore, the system must be able to generate a continuously growing texture

with appropriate resolution. Thus, instead of trying to generate a single mesh from all the depth maps, which is no trivial task and usually includes manual work for good quality, we fuse the scans *in screen space*. Since the object is movable, the projector's view of the object hardly ever matches the original scan perspective. Therefore, in every frame, a new *synthetic* view is generated by rendering each scan from the projector's current perspective and fusing overlapping fragments to estimate the real depth (see Sect. 5). Similarly, the paint interface shows a per-frame fused view of the scanned object that the user can inspect freely by moving the virtual camera.

The color input from the user is added by painting the fused view (see Sect. 6). The paint is reprojected into the existing scan views, which from then on include not only depth, but also color information. Hence, the fusion process fuses not only the scans' depth maps but also their colors. One can think of this procedure this way: By retroactively incorporating color into the scans, we play act as if this color had been scanned too. Thus, when projecting, we simply add colors that should have been there already. Fig. 3 visualizes the algorithm.

4 Scanning

Since we assume no prior knowledge about the target surface, the system starts with a scan to estimate the surface depth and normals. A suitable 3D scanning method for our interactive system must meet the following requirements:

1. Quality: Geometry-aware (multi-)projection mapping requires high-quality surface data to accurately compute per-pixel luminance. Otherwise, errors manifest themselves as lighting artifacts to which the human eye is sensitive.

2. Speed: For complex target objects we must assume that many scans are necessary to capture the surface entirely. Therefore, the scanning method must be reasonably fast to avoid long waiting periods for the user.

The depth stream from the RGB-D camera yields fast geometry estimates, however the images are very noisy and do not satisfy the quality requirement. Structured light scans, on the other hand, provide sufficient quality and our setup already provides the necessary projector and RGB camera.

4.1 Reconstruction

We employ grayscale Chebyshev Phase Shifting [13] with ten images. The first image is a white image and used to help filter foreground from background. Only pixels that exceed a certain brightness threshold are considered foreground and reconstructed. The remaining nine images encompass three phase shifts of three pattern frequencies each. In contrast to a Gray code scan for example, the number of images is constant and does not depend on the projector's resolution. The ten images are projected, one after another, and each projection is recorded by the camera. Unwrapping the phase from the camera recordings yields correspondences between projector frame columns and camera pixels. Since camera and projector are calibrated, three-dimensional points are created from these correspondences by computing plane-ray intersections. These 3D values are saved as an extended depth map from the camera's perspective. We additionally fill small holes in the scan for a more robust result.

Chebyshev Phase Shifting generates good-quality scans, however – like all structured light scans – not consistently across the entire surface. The quality decreases, the further the surface faces away from the camera. We introduce two thresholds t_h and t_l to handle such cases. If the angle between surface normal and camera direction exceeds t_h, we discard the depth value for being untrustworthy. Otherwise, if the angle still exceeds the lower threshold t_l, we keep the depth value, but mark it as unreliable. Reliability is stored as a value between 0 and 1, linearly interpolating between t_h and t_l.

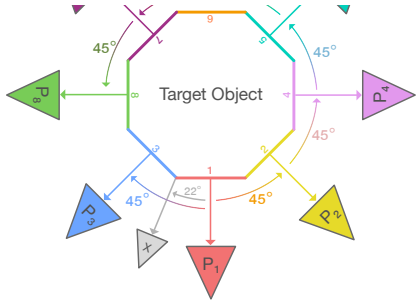


Figure 4. For a rotation trigger of 45° the angle between the current camera view and the closest existing scan view must exceed 45° to trigger a new scan.

Filled holes receive a reliability of 0. Unreliable areas trigger new scans (see Sect. 4.2.3) and are treated specially when fusing them (see Sect. 5).

4.2 Scan Trigger

A single structured light scan does not cover the entire surface of a three-dimensional target object. If the object is moved, e.g. rotated, the projection mapping system is quickly faced with new surface areas to which no scan data is available. Therefore, it is necessary to issue another scan from this new perspective.

Triggering a new scan should not be the responsibility of the user. If so, when moving the real-world object, the user would need to keep in mind, how much of the target was already scanned (and the corresponding quality) and stop the movement precisely there, where a new scan is necessary. The more the depth camera loses sight of previously scanned areas, the less reference points are left to determine the scans' matching rotation and translation. Failing to know the correct moment when to stop moving would thus lose the object's tracking. Furthermore, re-scanning already captured areas should also be prevented. Triggering a new scan must therefore be an automated process to be viable.

A trigger should not fire too early, to prevent too many interrupting structured light scans that provide only little new knowledge about the surface. However, waiting too long limits the surface that is available for projection mapping and risks losing the tracking.

We define object rotation and translation as the two main ways of exposing new surface to the system. We therefore propose both a rotation and a translation scan trigger. An additional quality trigger ensures sufficient overall quality.

4.2.1 Rotation Trigger

Each already existing scan is tracked based on the depth stream of the camera. Once the existing scans have rotated too far, a new scan is triggered. We therefor determine, in every frame, the angle between the camera and the tracked scans. The scan exhibiting the smallest angle is defined as the rotation-wise closest scan. Once this angle exceeds a pre-defined threshold, we assume that the camera's current view of the object is ideal for a new structured light scan, since even the closest scan is turned too far away for the current perspective. Fig. 4 depicts an example with a rotation threshold of 45°. After the first automatic scan from scan view P_1, the object is rotated. A second scan is triggered at P_2, since the angle between the camera and P_1 exceeds the threshold of 45°. Rotating in the other direction, it takes a total of 90° (45° + 45°), until the angle between the new position P_3 and the closest existing scan view (P_1) exceeds 45° again. The gray perspective X does not trigger a scan, since the threshold is not reached; P_1 is too close.

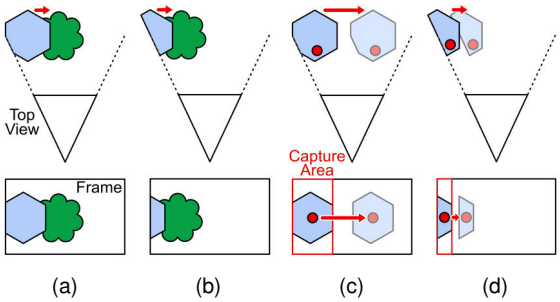


Figure 5. (a, b): Different relative surface gain between two consecutive scans (blue, green), triggered by translating the object to the right from different start positions. (c, d): Applying the same threshold ratio of 1.5, a larger scan and thus capture area allows more translation of the object before a new scan is triggered.

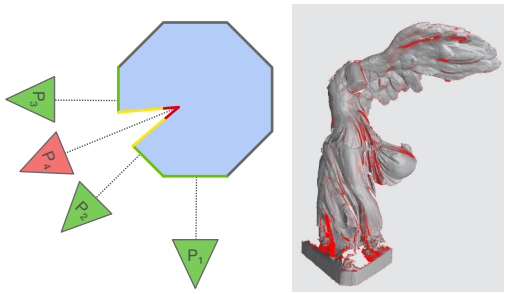


Figure 6. (a) The scan quality decreases if the surface normal points too far away (yellow) from the scanning poses (P_{1-3}) or if affected by self-shadows (red). An additional scan (P_4) is required to improve quality. (b) Fused view of two scans, taken 30° apart, depicting many unreliable areas (red).

4.2.2 Translation Trigger

If the target object is bigger than the camera's view frustum, rotation alone cannot expose the entire surface. Therefore, a translation trigger observes the object's translation with respect to the camera. Since translation measurements change significantly with the scene's scale, we apply a relative 2D measure by projecting the scans' centers of gravity (COG) into the camera frame. Similar to the rotation trigger we find the translation-wise closest scan, i.e., the scan that was translated the least from its original scan position. If even this closest distance, measured in camera pixels, exceeds a threshold, we once again assume that the camera's current view is ideal for a new structured light scan. However, in contrast to the rotation trigger, this threshold is not a constant, since the dimensions and placement of the object should influence the trigger's sensitivity. For example, in Fig. 5 the target object is translated to the right by the same amount in (a) and (b), triggering two scans (blue and green). Since, at the beginning, the object is located farther outside of the camera's view frustum in (b) than in (a), the first scan is smaller. In relative terms, the second scan thus provides a larger surface gain in (b). Considered the other way round, the translation trigger should trigger earlier in (b) than in (a) to keep the relative gain approximately equal.

To add a relative surface measure, we define the *capture area* to be the box centered around a scan's COG and stretching to the camera frame's closest vertical and horizontal border (see Fig. 5 (c) and (d)). The translation trigger then issues a new structured-light scan, if the closest scan's translation distance (component-wise) exceeds a threshold-defined *ratio* of the capture area's size. The capture area equals a scan's bounding box, if the object lies partially outside the camera frustum. If it lies completely inside, the capture area covers a larger area than the bounding box, since it includes the distance to the camera frame borders. That way, a small object centered in the frame does not trigger new scans too easily if translated only a little.

4.2.3 Quality Trigger

The quality of a structured light scan depends on the target surface and the angle from which it is scanned. If either camera or projector observe/project onto the surface from a too shallow angle, the scan's quality is diminished. Self-shadows even prevent parts of the surface from being scanned entirely. In Fig. 6a, three scans are issued from three different perspectives P_{1-3}, each 45° apart. Although a rotation trigger of 45° is ideal for the rest of the surface, the notch is reconstructed poorly. Thus, we implement a quality trigger that issues a new scan if poor quality is detected.

Since we base our projection not on a single mesh, but the image-based *fusion* of multiple scans, the quality similarly has to be determined by assessing the fused result. Therefore, in every frame, we render (see Sect. 5) the scene from the camera's perspective. With FragmentFusion [19], each pixel is a weighted average of the fragments projecting onto this pixel. We harness this behavior for quality assessment and likewise fuse the reliability value of each fragment, resulting in a confidence map (see Fig. 6b). Based on its confidence value, each fused pixel is classified as either good-quality (> 0.9) or bad-quality. If the fused frame's ratio of bad-quality pixels exceeds a pre-defined quality threshold, we trigger a new scan.

This quality trigger is based on the assumption that any part of the surface could be recorded better, if the camera's angle was more perpendicular to it. However, there are cases when a new scan from this presumably ideal perspective does not yield better geometry information (e.g., due to extreme surface normals from any perspective). In that case the quality trigger would issue new scans indefinitely, since the quality check always fails. Therefor we limit the trigger to fire only if there does not yet exist a scan that was recorded from approximately the current perspective.

4.3 Movement Detection

Our system expects some kind of interaction with the target object, e.g., a hand rotating the object, such that projections from another perspective are possible. As new scans are initiated automatically (see Sect. 4.2), extra care has to be taken to not include the user into the scans. Even if no human interacts with the object, e.g., if the object is put on a turntable, it must be guaranteed that the object itself is not in motion when a new scan starts. Therefor we perform a simple movement detection by analyzing the camera's depth image (the color image is affected by the projection). In contrast to many other common use cases, we are not interested in the kind of movement, movement direction or other advanced detection techniques. Instead, it is sufficient to know only if or if not there is movement – is a new scan possible right now or not?

We assume that motion manifests itself as change in the camera pixels' depth values, especially at silhouette borders. The Normalized Cross Correlation (NCC) is a common tool in image processing for quantifying change between two images A and B:

$$r = \frac{\sum_i (a_i - \bar{a})(b_i - \bar{b})}{\sqrt{\sum_i (a_i - \bar{a})^2 \sum_i (b_i - \bar{b})^2}} \quad (1)$$

where a_i and b_i are the image values of A and B at pixel index i and $\bar{a}$, $\bar{b}$ their arithmetic means respectively. By computing the NCC of two consecutive camera depth frames we measure how similar they are. If the similarity lies below a threshold t_m, we assume movement in the scene and wait until it stops before starting a new scan. If a scan had already started, the scan is interrupted and discarded.

The threshold t_m depends very much on the noisiness of the depth image. We compute an optimal threshold automatically by analyzing the noise level at startup. For the first few frames, we assume that the scene is motionless. We compute the average NCC μ and standard deviation σ and compute t_m as follows:

$$t_m = \mu - k \cdot \sigma \tag{2}$$

where k is a parameter that can be adjusted by the user. In practice, $k = 1$ is a good default value. Since the camera's noise may be affected by the image content (e.g., the object's distance to the camera), the threshold is recomputed periodically if no movement is detected. To further reduce incorrect movement detection, caused by sudden spikes in the noise level, we smooth the per-frame NCC exponentially.

Even if movement detection fails and a scan is issued, although the scene is still in motion or otherwise disturbed (e.g., if the user's hand still touches the target object), there is a last automatic check to ensure the scan's overall quality. The pose estimations of the reconstructed depth maps already roughly align to the real-world pose. In order to improve the tracking data, we additionally employ the Point-to-Plane Iterative Closest Point (ICP) algorithm to fine align the depth maps. If the resulting ICP error is too large, we assume that the scan failed due to unknown reasons and discard it.

5 Rendering New Perspectives

In dynamic projection mapping the target object can be moved around freely and the projection must update accordingly. Therefore, the projection mapping content must be rendered from the projector's perspective and be adjusted to the physical surface properties (e.g., surface normal). We employ geometry-aware projection mapping [11, 22], which requires as input the per-pixel surface normal and depth as well as the target color. Since our system scans the target object's surface automatically, we are provided with many reconstructed depth maps from various perspectives. Since the projection mapping algorithm works in screen space, it is not necessary to reconstruct a single mesh from them, but to only compute the per-pixel surface normal and depth from the projector's current perspective. The new perspective is already given by tracking the object and adjusting the scans' model matrices accordingly. In theory it would then be enough to render the scans with a simple depth test. In practice, however, camera noise and small calibration errors add up to noticeable depth discontinuities where the scans touch.

To overcome this issue, we implement FragmentFusion by Rückert et al. [19] that replaces the depth test by a fusion process. Their screen-space algorithm is based on the assumption that fragments with a similar depth describe the same surface. Instead of having one fragment 'win' a depth test over another, fragments are fused, i.e., form a weighted average. The scans are therefor consecutively rendered into the same framebuffer. Each new fragment f is compared to the current surface estimate p at the same position in the buffer. A fragment is discarded if it lies in the background or replaces the current surface estimate if it lies in the foreground. The decision is based on a per-pixel depth-dependant truncation distance. Otherwise, the fragment is fused into the surface, thereby updating the estimate for the surface point:

$$d_p \coloneqq \alpha d_p + (1-\alpha) d_f \tag{3}$$

$$c_p \coloneqq \alpha c_p + (1-\alpha) c_f \tag{4}$$

where d_p, d_f describe the depth and c_p, c_f the color of f and p respectively. The factor α determines the influence of a new fragment:

$$\alpha = \frac{w_p}{w_f + w_p} \tag{5}$$

$$w_f = \frac{cos\Theta}{d_k} \tag{6}$$

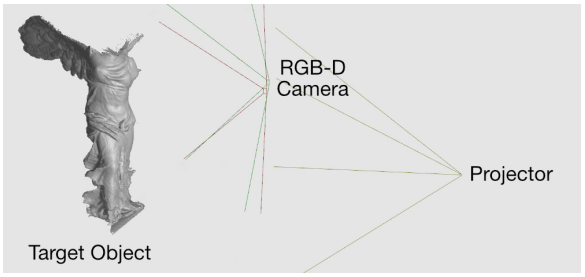


Figure 7. The user's input interface is a digital replicate of the real-world setup (see Fig. 2).

where d_k describes the fragment's reconstructed depth and Θ the angle between its original scan perspective and the rendering camera. w_p, initialized with 0, is updated with each fragment by adding w_f.

We extend the authors' algorithm by additionally interpolating fragment normals $\vec{n}_{\{p,f\}}$ and reliabilities $r_{\{p,f\}}$ (see Sect. 4.2.3):

$$\vec{n}_p \coloneqq \text{normalize}(\alpha \vec{n}_p + (1-\alpha)\vec{n}_f) \tag{7}$$

$$r_p \coloneqq \alpha r_p + (1-\alpha) r_f \tag{8}$$

By incorporating the reliability parameter into the weight computation we gain direct access to the fusion process:

$$w_f = \frac{cos\Theta}{d_k} \cdot r_f \tag{9}$$

The penalty of unreliable regions is thus increased; artificially filled holes are even ignored entirely if more reliable data is available from a second fragment.

The resulting values in the fusion buffer can then be used for additional deferred rendering, e.g., shading. The final depth, normal and color values are forwarded to the projection mapping system [11,22], which adjusts the target color's luminance to the surface geometry. The overall confidence result triggers new scans if needed (see Sect. 4.2.3).

Similarly to computing the projection output as described above, tracking the object also requires rendering the scans. Depth-based tracking compares the depth image of the depth camera with a rendered depth image from the same perspective, usually in form of the ICP algorithm. Thus, before we render the projection image, which requires tracking information, we render the scans from the depth camera's perspective and update our tracking state.

6 Projection Painting

The system described so far scans the target object on demand and per-frame fuses the scans for new synthetic perspectives that are then projected onto the target object. There is, however, no way to introduce content to be projected yet (besides a default shading). Usually, projection mapping content requires human input, e.g., an artist's design that is custom-made for the specific object. Since we never reconstruct a single mesh from all available scans, user input follows the same screen-space approach as the rest of the system.

6.1 Paint Texture

We provide the user with a paint interface, that shows the entire 3D scene as scanned so far. Thus, the scene acts as a digital twin to the real-world setup (see Fig. 7). Each frame, the scene is rendered with the FragmentFusion algorithm (see Sect. 5) from the user's desired perspective. The user paints the target object by drawing into a 2D texture that is always located in front of them. One can roughly think of this *paint texture* as transparent foil (see Fig.8). For rendering the active stroke on the object, the deferred shading mixes each fused fragment's base color with the paint texture's color at the same screen coordinate and applies the desired shading.

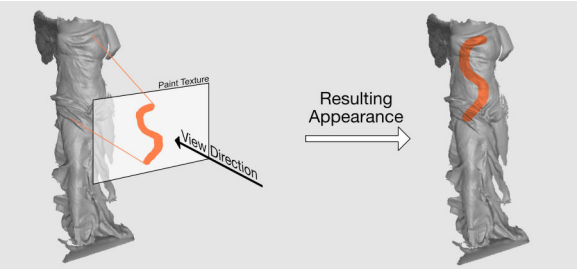


Figure 8. The user paints the paint texture. The color is applied by looking up each fragment's color in the paint texture at the same screen coordinate, giving the impression of a painted surface.

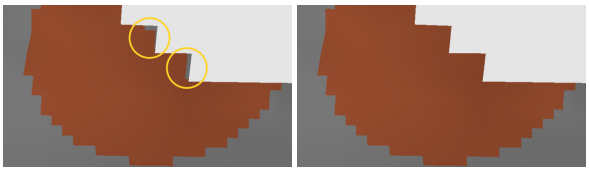
6.2 Paint Reprojection

Once the user changes their view frustum, the fragments' screen coordinates change and do not match the paint texture anymore, as they only align at the original paint perspective. In theory, we could collect all strokes from the exact same perspective into a single paint texture and remember the perspective. For each new perspective we would then allocate a new paint texture. During deferred rendering, we would project each fragment into every former paint frustum (in chronological order to achieve the correct layered effect) and sample its paint texture to retrieve the correct color. In practice, however, creating a new paint texture every time the user changes their perspective would quickly allocate a huge amount of memory and the system would run out of memory sooner or later.

We therefore propose thinking the FragmentFusion algorithm backwards. By reprojecting the paint into the virtual camera images, that correspond to our scan views, we benefit from the rendering's color fusion process. A user can only paint a surface that has already been scanned. Hence, any fused fragment being painted is derived from at least one scan. By reprojecting the color into all scans contributing to this fragment, we ensure that any fused result from them exhibits the desired color as well. To reproject the color, we render each scan from its original scan perspective. For each rendered fragment we determine the paint texture's corresponding texture coordinate by projecting the fragment's world position into the user's current paint frustum and sample its color. The rendering result is stored in each scan view's virtual camera image. Note that storing the scans' colors image-based, similar to the paint texture, does not require any three-dimensional parametrization, since the color lookup is not based on vertices' texture coordinates, but on a projection into the original scan view frustum.

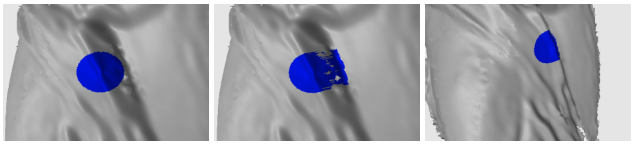
7 Implementation Details

A structured light scan provides us with discretely sampled 3D points of the surface. In contrast to an arbitrary point cloud, the points each originate from a single different camera pixel. Thus, from the camera's point of view, these discrete samples appear continuous and can be stored as a depth map. However, once we observe the depth map from a different perspective, e.g., when synthesizing a novel view, we are faced with holes that inevitably erupt when sampling such a discrete data structure. We therefore interpolate the depth values by generating a triangle mesh from each scan. The camera frame local pixel neighborhood yields the required topology. We currently construct each mesh once on the CPU after a successful scan. For a more robust result, we fill small holes in the mesh using an advancing front method (AFM) and improve the overall mesh quality by decimating the total triangle count and smoothing the surface. While the resulting speed of our CPU-based approach is sufficient for our meshes, faster approaches are possible. Since the entire algorithm operates in screen space and requires depth value interpolation only during rendering, it would also be possible to add the mesh construction as a compute shader preprocessing step.



(a) Without Dilation (b) With Dilation

Figure 9. Close-up of the same paint stroke on the fused surface with and without prior dilation of the reprojected paint. Small areas (circled in yellow) remain unpainted when skipping dilation.



(a) (b) (c)

Figure 10. (a) Correctly displayed paint stroke, if the virtual camera image's perspective matches the original scan's perspective. (b) The image is incorrectly stored from the projector's point of view. The different perspectives for geometry and color cause stains that (c) only from the projector's point of view (wrongfully) appear to be correct.

To prevent painting on the back of the object, we perform a position check similar to shadow mapping. A virtual camera image's fragment is only painted if its relative world position towards the paint perspective matches the user view's fusion buffer. We perform a world position check instead of a depth check to ease aliasing artifacts like paint acne (similar to shadow mapping's shadow acne) and help tiny creases to be painted instead of skipped.

A new scan starts with an empty virtual camera image. When fusing the base color, we filter any non-existing color information to prevent new scans from mixing invalid color into the fragment.

For rendering the active paint stroke Sect. 6.1 describes that a fragment's color is sampled from the paint texture at the same screen-space coordinate, since the render and paint perspective match. This is not the case for the projection, since the projector's perspective is independent from the user's. To render the active paint stroke from a projector's perspective, each fragment must first be reprojected into the viewer's view frustum before that paint texture is sampled.

Another issue that arises when reprojecting a texture into another is quantization errors. Hence, when reprojecting the paint texture into a virtual camera image, small unpaintable areas are introduced, especially around holes and at borders. To prevent this effect, we dilate the reprojected paint by a few pixels before storing it in the virtual camera image (see Fig. 9).

In Sect. 6.2 we explain that the paint texture is reprojected into the scans and stored as virtual camera images. One might think that the virtual camera images' view frusta could be located anywhere (sensible) in the scene, as long as the same perspective is used for sampling it later when fusing. However, it is important that a virtual camera image's render frustum equals the scanning RGB camera's frustum. Since the object's geometry is reconstructed from the RGB camera's point of view, we would suffer from quantization errors and self-shadows, if we were to sample the color from a different perspective. The perspectives of reconstructed depth and color must match. Fig. 10 shows exemplarily, how a surface at small grazing angles is painted wrongly if the virtual camera image is located at the projector's point of view (similar to the camera's perspective, but not the same).

8 Evaluation

In our setup we employ an `Intel Realsense SR300` RGB-D camera (depth image: 640×480 pixels, RGB image: 1280×720 pixels at 60 FPS or 1920×1080 pixels at 30 FPS) and an `NEC ME382U` projector (1920×1200 pixels) at 60 FPS. Our target objects are a 31 cm high 3D print of the *Winged Victory of Samothrace* statue and several other ordinary items (e.g., shoes). Computations and rendering are performed on an `Intel Core i7-6700K` (4.00 GHz) CPU, 32 GB of RAM and an `NVIDIA GeForce GTX 1080` graphics card. Calibrating this setup (independently of the target object) takes about $1-2$ minutes and is only necessary if the relative pose of camera to projector is changed. We prevent this by mounting both on a cage (see Fig. 2).

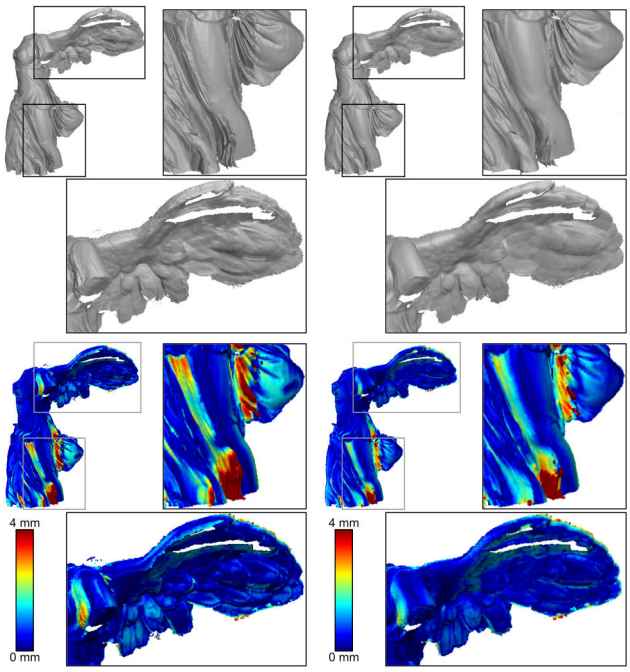
8.1 Scanning

A new scan is triggered, if the target object is rotated or translated far enough or if the existing quality is not sufficient. Since all of our test objects possess a clearly defined top and bottom, we tested the rotation trigger only along the up-axis. However, since the core problem merely is finding the rotation-wise closest existing scan, our approach is applicable to three-dimensional rotations as well. To prevent any user or auxiliary objects to be captured by the scan, we check that there is no movement in the scene before scanning. However, we noticed that users tend to pause for a moment, after having moved the object, before stepping away. We therefore included a 1 second long waiting period combined with a red warning projection to signal users that a new scan starts shortly. Please see the supplementary video for a demonstration of the scanning process.

The type of trigger issuing a new scan depends on the configured thresholds and the shape of the target object. For small objects that fit into the camera frustum we recommend a rotation threshold of 30° to 45°, which we also applied to most of our tests. For larger objects the threshold should be lower. The many folds of the *Winged Victory of Samothrace* statue cause frequent self-shadowing during scanning. Therefore, it is almost exclusively the quality trigger (set to 5 %) issuing new scans when rotating the statue. For other small-sized objects we tested, the rotation trigger fires in about one third of cases. This difference shows that the number of scans required to cover the entire target object strongly depends on the object's shape. For the *Winged Victory of Samothrace* statue and a 45° rotation threshold it took seventeen scans on average to fully rotate it by 360°. In contrast, using the same settings, a more regular shaped motorcycle helmet required only twelve scans, since the quality trigger fired less often. If not translating a small object from one side of the camera frustum to the other on purpose, the translation trigger never triggers. The natural position of the object lies at the center of the camera frustum and the thus large capture area demands significant translation to trigger a new scan. This behavior is by design, since small objects expose new surface only by rotation and additional scans caused by translation are redundant. With larger objects, it is exclusively the translation trigger that fires upon translation and never the quality trigger. This seems reasonable since standard camera lenses see roughly the same of a surface regardless of the position within the frustum, hence never observing a better quality. We assume that the quality trigger could indeed complement the translation trigger upon translation if using fisheye lenses.

8.2 Rendering

Instead of depth testing we apply FragmentFusion to render multiple scans that (partially) cover the same surface. The benefit from FragmentFusion over depth testing becomes especially apparent, the more the scans overlap. An object particularly difficult to scan, due to the many cloth folds and details, is the *Winged Victory of Samothrace* statue, since its surface normals vary a lot. We scanned its left side six times with a sensitive rotation threshold. The individual scans look similar to the human eye, but when rendering all of them with depth testing enabled, we see artifacts caused by depth discontinuities that look like multiple layers on top of each other (see Fig. 11a). FragmentFusion instead smooths out most of these artifacts and the result is more pleasing to the eye (see Fig. 11b). The bottom row of Fig. 11 compares both render results with the original full mesh (ground truth) of the statue by applying the Hausdorff distance. Since both rendered geometries are ICP-fitted into the ground truth mesh – similar to tracking the real-world object – the overall Hausdorff mean is equivalent: 0.679 mm (Depth Test) vs. 0.643 mm (FragmentFusion). FragmentFusion however reduces the Hausdorff distance at critical areas where the depth deviation is high. The Hausdorff images also depict a more homogeneous error distribution (the less features are visible, the better), which makes errors less noticeable.



(a) Depth Test (b) FragmentFusion

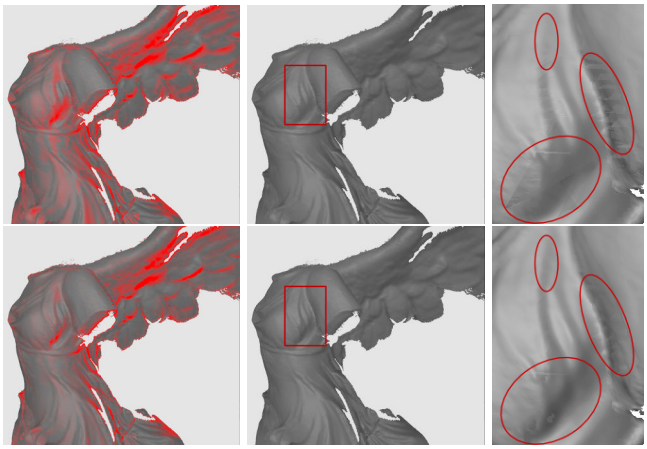
Figure 11. Six scans rendered with (a) depth testing and (b) FragmentFusion. The top row depicts a default color shading, the bottom row the Hausdorff distance to the ground truth mesh. FragmentFusion reduces the layering effect and the quality is more homogeneous.

We extend the original FragmentFusion algorithm by factoring the reliability of a fragment into the weight computation (see Sect. 5). The effectiveness becomes visible mostly in critical areas of low confidence (see Fig. 12). The overall confidence in the confidence map increases and some unreliable areas become more defined, if more reliable surface data is available. In Fig. 12c, the folds are sharpened and artifacts diminished.

8.3 Painting

The interactive painting results are best seen in the supplementary video. Fig. 13 shows painting results for three different target objects. The paint texture has a resolution of 1920×1200 pixels and the virtual camera images 3840×2400 pixels each. The drawings in (a) and (c) were painted directly in our application. For the shoe, the user first made several scans, then used the recorded RGB images as reference, while painting the design with *external* 2D paint software. Due to the matching perspectives of painting and scan, these 2D drawings could directly be loaded as the virtual camera images.

An advantage of our system over regular painting pipelines is the



(a) Confidence (b) FragmentFusion (c) Details

Figure 12. Impact of the reliability factor. Top row without, bottom row with confidence weighting. (a) The overall unreliability in the confidence map decreases (smaller red areas). (b) Some folds become more defined and artifacts are mitigated. The highlighted area is magnified in (c) and contrast-adjusted for better visibility.

integration of paint and projection functionality into the scanning system itself. Although still possible, the user is not forced to first scan the entire object and only then paint it. Since our system scans on-demand whenever a new surface part is exposed, the target object is paintable after the first recorded structured light scan.

Although we demonstrate the results with a single projector, our system is capable of multi-projection mapping. FragmentFusion allows rendering the scans from any novel perspective. Thus, a second projector would retrieve its own color and geometry buffer. All projector framebuffers are then luminance-adjusted by the multi-projection mapping algorithm [11, 22].

9 Lessons Learned and Limitations

In our current implementation the user has to stop moving the object, once a new scan is triggered. Although we tried to keep the interruption short by using a phase shifting algorithm that requires only few frames, the interruption does interfere with the intended movement. There are structured light scanning methods that capture even moving objects, e.g., Rusinkiewicz et al. [20], however extra care has to be taken that the user is not included into the reconstruction. Additionally, the perspectives of the reconstructed geometry and the virtual camera images must match to avoid artifacts (see Fig. 10).

Our system allows for horizontal and vertical translation of the target object, but not for significant movement into the back- and foreground, e.g., when moving a large object further into the distance, thereby exposing more of its surface. To support such use cases, the translation trigger must be adapted. However, we did not study how well the FragmentFusion algorithm handles the fusion of depth maps with different resolutions.

We track all scans by observing the target object through the camera's depth stream. As a consequence, we do not need to attach markers or tracking hardware to the object, which makes the system more accessible. However, depth-based tracking tends to be noisy (at least with consumer-grade hardware) and requires the object to exhibit distinct depth features. Using our system as described, a user could not paint the entire surface of rotationally symmetric objects like a ball or vase, since the first scan would stick to the front and not rotate with the object. Futhermore, our tracking does not support deformation or changing the layout among multiple objects.

We fill small holes in the reconstruction to improve the quality. However, hole filling should be performed with caution. If the filler depth is significantly smaller (i.e., closer) than the original depth, it overlays a reliable scan of the hole region, since FragmentFusion classifies the filler depth as foreground and discards the rest.

We designed our system for painting white objects – similar to drawing on a white sheet of paper. For painting colored objects, the projection image must additionally balance out the surface color.

10 Conclusion

We presented a system that combines 3D scanning, FragmentFusion and projection mapping to allow for spontaneous projection painting. Automatic triggers issue new structured light scans, once the target object is moved far enough to expose undiscovered surface areas to the system. The reconstructed depth from all scans is fused in screen space to generate novel synthetic views for the projector by applying the FragmentFusion algorithm, similar to image-based rendering. The user adds color to the projection by painting synthesized views from a second, user-controlled perspective. The color is retroactively incorporated into the scans and included in the fusion process.

For future work, we see great potential in improving the tracking mechanism, as well as adding a more dynamic user input method.

Acknowledgments

The original 3D model of Winged Victory of Samothrace (`http://www.thingiverse.com/thing:196038`) by CosmoWenman is licensed under the Creative Commons - Attribution license. `http://creativecommons.org/licenses/by/3.0/`

References

[1] H. Asayama, D. Iwai, and K. Sato. Fabricating Diminishable Visual Markers for Geometric Registration in Projection Mapping. *IEEE Transactions on Visualization and Computer Graphics*, 24(2):1091–1102, Feb. 2018. doi: 10.1109/TVCG.2017.2657634

[2] D. Bandyopadhyay, R. Raskar, and H. Fuchs. Dynamic shader lamps : painting on movable objects. In *Proceedings of the IEEE and ACM International Symposium on Augmented Reality*, pp. 207–216, 2001. doi: 10.1109/ISAR.2001.970539

[3] A. H. Bermano, M. Billeter, D. Iwai, and A. Grundhöfer. Makeup Lamps: Live Augmentation of Human Faces via Projection. *Computer Graphics Forum*, 36(2):311–323, May 2017. doi: 10.1111/cgf.13128

[4] A. Grundhöfer and I. Daisuke. Recent Advances in Projection Mapping Algorithms, Hardware and Applications. *Computer Graphics Forum*, 37(2):653–675, 2018. doi: 10.1111/cgf.13387

[5] S. Izadi, A. Davison, A. Fitzgibbon, D. Kim, O. Hilliges, D. Molyneaux, R. Newcombe, P. Kohli, J. Shotton, S. Hodges, and D. Freeman. KinectFusion: real-time 3D reconstruction and interaction using a moving depth camera. In *Proceedings of the 24th annual ACM symposium on User interface software and technology - UIST '11*, p. 559. ACM Press, Santa Barbara, California, USA, 2011. doi: 10.1145/2047196.2047270

[6] B. Jones, R. Sodhi, M. Murdock, R. Mehra, H. Benko, A. Wilson, E. Ofek, B. MacIntyre, N. Raghuvanshi, and L. Shapira. RoomAlive: Magical Experiences Enabled by Scalable, Adaptive Projector-camera Units. In *Proceedings of the 27th Annual ACM Symposium on User Interface Software and Technology*, UIST '14. ACM, NY, USA, 2014. doi: 10.1145/2642918.2647383

[7] B. R. Jones, H. Benko, E. Ofek, and A. D. Wilson. IllumiRoom: Peripheral Projected Illusions for Interactive Experiences. In *ACM SIGGRAPH 2013 Emerging Technologies*, SIGGRAPH '13, pp. 7:1–7:1. ACM, New York, NY, USA, 2013. doi: 10.1145/2503368.2503375

[8] P. Kurth, V. Lange, C. Siegl, M. Stamminger, and F. Bauer. Auto-Calibration for Dynamic Multi-Projection Mapping on Arbitrary Surfaces. *IEEE Transactions on Visualization and Computer Graphics*, 24(11):2886–2894, Nov. 2018. doi: 10.1109/TVCG.2018.2868530

[9] V. Lange, P. Kurth, B. Keinert, M. Boss, M. Stamminger, and F. Bauer. Proxy Painting: Digital Colorization of Real-world Objects. *Journal on Computing and Cultural Heritage*, 13(3):1–20, Aug. 2020. doi: 10.1145/3377145

(a) (b) (c)

Figure 13. Projection (painting) results on (a) a motorcycle helmet, (b) a shoe, and (c) the *Winged Victory of Samothrace* statue. The white projections are shaded fusion results of successfully reconstructed areas.

[10] V. Lange, C. Siegl, M. Colaianni, P. Kurth, M. Stamminger, and F. Bauer. Interactive Painting and Lighting in Dynamic Multi-Projection Mapping. In *Augmented Reality, Virtual Reality, and Computer Graphics. AVR 2016*, pp. 113–125. Springer, Cham, 2016. doi: 10.1007/978-3-319-40651-0_10

[11] V. Lange, C. Siegl, M. Colaianni, M. Stamminger, and F. Bauer. Robust Blending and Occlusion Compensation in Dynamic Multi-Projection Mapping. In *Eurographics 2017 - Short Papers*, 2017. doi: 10.2312/egsh.20171000

[12] L. Miyashita, Y. Watanabe, and M. Ishikawa. Midas projection: Markerless and modelless dynamic projection mapping for material representation. *ACM Trans. Graph.*, 37(6), Dec. 2018. doi: 10.1145/3272127.3275045

[13] D. Moreno, W. Y. Hwang, and G. Taubin. Rapid Hand Shape Reconstruction with Chebyshev Phase Shifting. In *2016 Fourth International Conference on 3D Vision (3DV)*, pp. 157–165, Oct. 2016. doi: 10.1109/3DV.2016.24

[14] G. Narita, Y. Watanabe, and M. Ishikawa. Dynamic Projection Mapping onto Deforming Non-Rigid Surface Using Deformable Dot Cluster Marker. *IEEE Transactions on Visualization and Computer Graphics*, 23(3):1235–1248, Mar. 2017. doi: 10.1109/TVCG.2016.2592910

[15] R. A. Newcombe, A. Fitzgibbon, S. Izadi, O. Hilliges, D. Molyneaux, D. Kim, A. J. Davison, P. Kohi, J. Shotton, and S. Hodges. KinectFusion: Real-time dense surface mapping and tracking. In *2011 10th IEEE International Symposium on Mixed and Augmented Reality*, pp. 127–136. IEEE, Basel, Oct. 2011. doi: 10.1109/ISMAR.2011.6092378

[16] H. Park, M.-H. Lee, S.-J. Kim, and J.-I. Park. Surface-Independent Direct-Projected Augmented Reality. In D. Hutchison, T. Kanade, J. Kittler, J. M. Kleinberg, F. Mattern, J. C. Mitchell, M. Naor, O. Nierstrasz, C. Pandu Rangan, B. Steffen, M. Sudan, D. Terzopoulos, D. Tygar, M. Y. Vardi, G. Weikum, P. J. Narayanan, S. K. Nayar, and H.-Y. Shum, eds., *Computer Vision – ACCV 2006*, vol. 3852, pp. 892–901. Springer Berlin Heidelberg, Berlin, Heidelberg, 2006. Series Title: Lecture Notes in Computer Science. doi: 10.1007/11612704_89

[17] R. Raskar, G. Welch, K.-L. Low, and D. Bandyopadhyay. Shader Lamps: Animating Real Objects With Image-Based Illumination. In *Proceedings of the 12th Eurographics Workshop on Rendering Techniques*. Springer-Verlag, London, 2001. doi: 10.1007/978-3-7091-6242-2_9

[18] C. Resch, P. Keitler, and G. Klinker. Sticky Projections-A Model-Based Approach to Interactive Shader Lamps Tracking. *IEEE Transactions on Visualization and Computer Graphics*, 22(3):1291–1301, Mar. 2016. doi: 10.1109/TVCG.2015.2450934

[19] D. Rückert, M. Innmann, and M. Stamminger. FragmentFusion: A Light-Weight SLAM Pipeline for Dense Reconstruction. In *2019 IEEE International Symposium on Mixed and Augmented Reality Adjunct (ISMAR-Adjunct)*, pp. 342–347. IEEE, Beijing, China, Oct. 2019. doi: 10.1109/ISMAR-Adjunct.2019.00-15

[20] S. Rusinkiewicz, O. Hall-Holt, and M. Levoy. Real-time 3D model acquisition. *ACM Transactions on Graphics*, 21(3):438–446, July 2002. doi: 10.1145/566654.566600

[21] H. Shum and S. B. Kang. Review of image-based rendering techniques. p. 2. Perth, Australia, May 2000. doi: 10.1117/12.386541

[22] C. Siegl, M. Colaianni, L. Thies, J. Thies, M. Zollhöfer, S. Izadi, M. Stamminger, and F. Bauer. Real-time Pixel Luminance Optimization for Dynamic Multi-projection Mapping. *ACM Trans. Graph.*, 34(6):237:1–237:11, Oct. 2015. doi: 10.1145/2816795.2818111

[23] V. Srinivasan, H. C. Liu, and M. Halioua. Automated phase-measuring profilometry: a phase mapping approach. *Appl. Opt.*, 24(2):185–188, Jan 1985. doi: 10.1364/AO.24.000185

[24] D. Tone, D. Iwai, S. Hiura, and K. Sato. FibAR: Embedding Optical Fibers in 3D Printed Objects for Active Markers in Dynamic Projection Mapping. *IEEE Transactions on Visualization and Computer Graphics*, 26(5):2030–2040, May 2020. doi: 10.1109/TVCG.2020.2973444

[25] Y. Watanabe, T. Kato, and M. ishikawa. Extended Dot Cluster Marker for High-speed 3D Tracking in Dynamic Projection Mapping. In *2017 IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*, pp. 52–61. IEEE, Nantes, Oct. 2017. doi: 10.1109/ISMAR.2017.22

[26] Y. Yasumuro, M. Imura, Y. Manabe, O. Oshiro, and K. Chihara. Projection-based augmented reality with automated shape scanning. p. 555. San Jose, CA, Mar. 2005. doi: 10.1117/12.586303

[27] S. Zhang. High-speed 3D shape measurement with structured light methods: A review. *Optics and Lasers in Engineering*, 106:119–131, July 2018. doi: 10.1016/j.optlaseng.2018.02.017

[28] Y. Zhou, S. Xiao, N. Tang, Z. Wei, and X. Chen. Pmomo: Projection Mapping on Movable 3D Object. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*, CHI '16, pp. 781–790. ACM, New York, NY, USA, 2016. doi: 10.1145/2858036.2858329

[29] M. Zollhöfer, P. Stotko, A. Görlitz, C. Theobalt, M. Nießner, R. Klein, and A. Kolb. State of the Art on 3D Reconstruction with RGB-D Cameras. *Computer Graphics Forum*, 37(2):625–652, May 2018. doi: 10.1111/cgf.13386