Sticky Projections - A New Approach to Interactive Shader Lamp Tracking

Christoph Resch[*] EXTEND3D GmbH
Peter Keitler [†] EXTEND3D GmbH
Gudrun Klinker[‡] TU München

Abstract

Shader lamps can augment physical objects with projected virtual replications using a camera-projector system, provided that the physical and virtual object are well registered. Precise registration and tracking has been a cumbersome and intrusive process in the past. In this paper, we present a new method for tracking arbitrarily shaped physical objects interactively. In contrast to previous approaches our system is mobile and makes solely use of the projection of the virtual replication to track the physical object and "stick" the projection to it.

Our method consists of two stages, a fast pose initialization based on structured light patterns and a non-intrusive frame-by-frame tracking based on features detected in the projection. In the initialization phase a dense point cloud of the physical object is reconstructed and precisely matched to the virtual model to perfectly overlay the projection. During the tracking phase, a radiometrically corrected virtual camera view based on the current pose prediction is rendered and compared to the captured image. Matched features are triangulated providing a sparse set of surface points that is robustly aligned to the virtual model. The alignment transformation serves as an input for the new pose prediction. Quantitative experiments show that our approach can robustly track complex objects at interactive rates.

1 Introduction

Projector based illumination and augmentation has become an important area in Spatial Augmented Reality (SAR) in the last decade, not least because powerful projector devices are getting smaller and cheaper. The usage of projectors in SAR in the past can be categorized mainly in two types. On the one hand, (mostly) planar objects are augmented with two-dimensional content such as text, image and videos to act as an innovative display. From early applications such as the "Luminous Room" [17] where each surface of the environment could possibly serve as an interactive display, this concept has undergone thorough research in terms of projection surface complexity, view-dependent rendering, tracking, geometric and radiometric calibration and compensation and multiple projector setups. Indispensable in all applications is one or several cameras to complement the projection system and allow for a close feedback-loop. Raskar's "iLamp" concept [13] for instance enabled clusters of mobile projector systems to create a seamless projection by using cameras that were mounted to the projectors.

However, there has been little research on augmenting real 3D objects. In contrast to 2D approaches that project arbitrary information which is not related to the projection surface, 3D methods explicitly try to alter the appearance of arbitrary, complex objects by projecting an aligned rendering of an exact virtual replication with modified texture on it. This *shader lamp* concept was first introduced by Raskar et. al. [14] and further worked out with respect to e.g. animation [15] or radiometric-realistic appearance [10]. Mostly designers, architects and artists profit from this technology, as they can - once a neutral "mockup" is produced - spatially visualize design variants without actually creating them physically.

Whereas the manual creation of those virtual models by scanning or manual tactile measuring seemed costly in the past and has discouraged people to work with this technology, the rapidly declining cost of 3D printers and their entrance to the consumer level have started to solve this problem since first the virtual model is designed and then physically replicated. In general we think that the concept of shader lamps is currently revived by affordable 3D printers and is a perfect addition as the mostly neutral white printer's output is well suited as a projection surface.

More than in any other AR modality such as HMDs it is of vital importance in the shader lamp concept that the rendered model precisely overlays the physical object - the illusion is quickly destroyed even with small misalignments. Thus a precise registration technique is crucial, that ideally is at the same time mobile and interactive and does not require any modifications to the scene. This paper deals with the problem of tracking such illuminated objects while they are moved - in particular with no auxiliary means than the projection itself. We motivate our approach in the following by quickly summarizing current registration approaches.

2 Previous work

In the original shader lamp [14] concept, a cumbersome static registration was performed by manually moving a cross hair in the projector view to pixels that illuminate known object points. In [4], standard square marker tracking was used to steer the projection to the right location. Bandyopadhyay [2] equipped the illuminated objects with infrared and magnetic sensors which allowed interactive tracking but suffered from low working distances. In *DisplayObjects* [1] the authors used a VICON tracking system to track the display objects. Even though the latter two examples allow the illuminated object to move freely, the projector and its tracking hardware are mounted at a fixed position. However, in many situations the reverse scenario is desired and a mobile projection system should move around the illuminated object. Furthermore, dedicated, professional tracking hardware is expensive and requires a sophisticated and complex calibration to the projection devices.

Tracking algorithms based on commercial, lightweight depth cameras such as the Kinect are suitable only to a limited extent, since the required pixel precise overlay in a shader lamp application suffers from the low resolution of the depth sensor [8]. One should still observe the development of such commercial products as they might be a sound alternative in the near future. Feature or model based tracking algorithms are also able to track objects for being correctly illuminated [11]. The obvious problem with such methods are the unfavorable lighting conditions in a projection scenario, where ambient light is usually low. Also, the projection itself can heavily interfere with the features found on the physical object.

More promising are methods that use the projector not only for augmenting the scene but also as an active light sensor to recover the projection surface and/or track the projector with respect to this

[*]e-mail: christoph.resch@extend3d.de
[†]e-mail: peter.keitler@extend3d.de
[‡]e-mail: klinker@in.tum.de

IEEE International Symposium on Mixed and Augmented Reality 2014
Science and Technology Proceedings
10 - 12 September 2014, Munich, Germany
978-1-4799-6184-9/13/$31.00 ©2014 IEEE

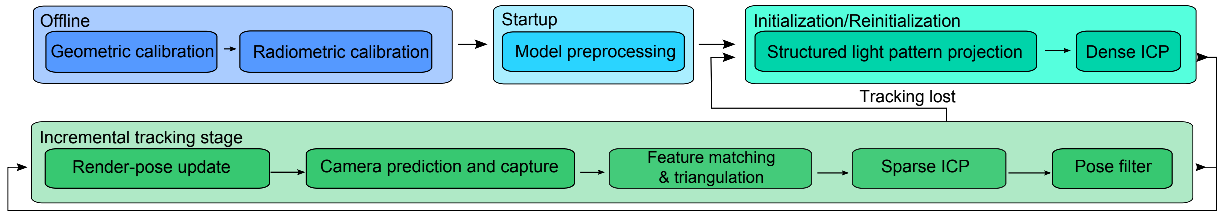

Offline
Geometric calibration
Radiometric calibration
Startup
Model preprocessing
Initialization/Reinitialization
Structured light pattern projection
Dense ICP
Tracking lost
Incremental tracking stage
Render-pose update
Camera prediction and capture
Feature matching & triangulation
Sparse ICP
Pose filter

Figure 1: Sequence diagram of the proposed method. Having calibrated the system geometrically and radiometrically once offline, the system samples the virtual model surface to get an accurate and occlusion-corrected point cloud at start up. In the initialization phase a dense point cloud of the physical mockup is acquired by means of structured light and the exact pose is computed by matching the sampled model point cloud with the point cloud of the scan. Once the virtual model is projected on the physical object, the continuous frame-by-frame tracking starts, in which first the illuminated object is captured by the camera. At the same time, the camera view is also rendered virtually, given the current object pose. Features are found in the predicted image and matched in the captured image. After triangulating the correspondences the sparse set of 3D feature points is aligned to the sampled virtual model via ICP and the transformation is used for correcting the new render pose. A filter ensures smooth pose updates over time. Should the incremental tracking fail, the user can trigger the initialization stage again.

surface.

When focusing on the approaches that do not interrupt projected user imagery we can distinguish between methods that inject imperceptible codes in the projection and methods that use projection-inherent features. For the former, Cotting et. al. [3] modified the modulation pattern of a DLP projector to embed a gray-code pattern sequence in the projected content that was used for reconstructing the projection surface. However, this approach required complex modifications to hardware to account for synchronization timings and resulted in a loss of image contrast. Also, spatial encoding techniques such as gray codes require multiple patterns to be projected over time during which the projector-surface relation must remain static.

Another way of tracking the projector is to use the user imagery itself. A very general, model-based approach was provided by Zheng et. al. [19] that iteratively optimizes the pose of the projected content to minimize the 2D image differences between the real and expected image. Unfortunately, they tested their projection-based scenario only with flat surfaces and not with a shader lamp typical complex object. Instead of using the whole image, concepts based on discrete features found in the camera and projector image were first introduced by Yang and Welch [18] and later refined by Johnson and Fuchs [7]. The latter setup consisted of a static camera and a mobile projector. Offline, the 3D coordinates of the projection surface for each camera pixel were computed by a stereo structured light method. Online, a feature matching algorithm provided a set of 2D-2D correspondences between camera and projector image and thus - with the previous offline calibration - indirectly a set of 2D-3D correspondences in projector space which allowed for tracking the projector continuously. Nevertheless, the camera as well as the projection surface had to remain static which is not suitable for a dynamic shader lamp scenario.

Our approach extends this method to work also in a scenario, where the object or the projection device can be moved arbitrarily around in space. We therefore carry out two modifications to [7]: Firstly, the camera is rigidly attached to the projector. Secondly, the projector pose estimation based on 2D-3D correspondences is replaced by a point cloud registration technique, using triangulated 3D points from the 2D-2D correspondences on the real surface and the geometry of the virtual model as a matching target. The incremental nature of our tracking approach enables interactivity close to real-time, not least because we are utilizing the computation power of the GPU.

3 Overview of our Approach

Our system consists of a standard video projector with a rigidly attached camera. Provided a physical object and a corresponding virtual 3D model our system can simultaneously track and correctly illuminate the object with the projection of the virtual model itself. Figure 1 depicts the flow chart of our method. Online, it consist of two phases: A fast initialization of the object pose and a continuous incremental tracking based on the projection of the virtual model only. If the tracking is lost, the initialization phase can be triggered again. Section 4 will explain this incremental concept in detail.

Offline, the camera-projector system must be calibrated once geometrically to compute the intrinsic and extrinsic parameters of the optical components. We calibrated our system with a planar calibration board approach similar to [12] that is projecting a white circle pattern interleaved with the black printed circles of the calibration board. Considering the OpenCV distortion model also for the projector, we reached a system calibration accuracy of 0.25 px reprojection RMS error. The radiometrical calibration described in [7] was carried out, too.

Finally, each virtual model that should be visualized on its corresponding physical object, must be preprocessed for the later registration step, which is done automatically when a new design is loaded. The preprocessing step is explained in section 4.3.

4 Incremental Tracking

The core component of our concept is the incremental tracking stage in which features of the projected texture are reconstructed (Section 4.1) and matched to the virtual model to update the model pose (Section 4.2). Although an incremental approach comes with inherent restrictions to fast object movement, it has the advantage of locality and allows us to make assumptions on the feature position and the surroundings in image and world space.

4.1 Feature extraction and triangulation

In the first part of our incremental tracking stage, we apply the feature extraction and correspondence estimation method of [7] to find matching points between the camera and projector image. Whereas in their approach the virtual content was an ordinary video stream, our content is the texture of the virtual model itself. As the feature detection is carried out for each frame separately, the texture of the virtual model can also be animated and need not remain static. After correspondences have been found by the Lucas-Kanade tracker, we reject wrong matches based on epipolar constraints and triangulate the remaining point pairs using the direct linear transform method proposed in [5]. These points might represent valid surface points that shall be registered to the virtual model in the next section.

4.2 Pose estimation by point cloud registration

Having access to a virtual replication of our real model by means of a geometric surface description allows us to use point cloud registration techniques to calculate a rigid transformation that aligns reconstructed feature points to virtual surface points. There has been an abundance of research on point cloud registration in the past in terms of local and global optimization, rigidness of objects, feature space and many more. Among them the *ICP* (Iterative Closest Point) algorithm has proven to be an efficient and fast registration method if a good initial relative pose is known, which is true in our case as our incremental tracking approach assumes small object movements in each frame so that the previous pose can serve as initial pose. One prominent example using the same local assumption is the *KinectFusion* algorithm [6] that also incrementally registers range images of the sensor in an ICP like manner.

Rusinkiewicz [16] has surveyed many variants of the ICP with respect to correspondence point-pair selection, weighting, assigning error metrics and minimizing them. The author noted that the point-to-plane metric converges orders of magnitude faster as it allows faster tangential movement. This is particularly important for flat surfaces where one surface must "slide" over another. Our registration algorithm is based on this particular ICP variant. Given a source and target surface with already estimated corresponding point-pairs (s_i, t_i), whereas for each target point additionally its surface normal n_i is known, the error ε that is minimized in each ICP iteration for the point-to-plane metric is

$$\varepsilon = \sum_i \langle [\mathbf{R}\ t]\ s_i - t_i, n_i \rangle \tag{1}$$

where $\langle , \rangle$ denotes the inner product and $[\mathbf{R}\ t]$ the transformation that minimizes ε in the current iteration. As shown by Low [9], the nonlinear minimization problem for Equation 1 can be approximated by a linear least squares solution if the expected rotation angles are small. Again, the above mentioned locality allows us to justify this approximation.

Although we already integrated several outlier detection mechanism in our feature matching stage, it might be possible that some 3D points still are wrongly triangulated and passed to the ICP iterations. To deal with such outliers inside the ICP iterations we used a simple RANSAC scheme where various randomly-selected subsets of the closest corresponding point pairs are evaluated using the above error metric and the optimal subset defines the inliers for the final transformation estimation.

At the end, single exponential smoothing, that incorporates only a certain fraction of the new pose in the current object pose, was implemented. Although smoothing reduces the responsiveness of the system we noted a considerable improvement in tracking stability, in particular with difficult virtual textures.

4.3 Model preprocessing

Usually the virtual model is not stored as a point cloud but instead as a triangulated mesh. Since our registration algorithm expects a point cloud with normal information, the virtual model must be preprocessed at start up. As our feature extraction algorithm may reconstruct possibly any point on the surface, simply using the face corners of the mesh is not enough - we need a dense sampling of the virtual mesh up to a certain accuracy ε that is dependent on the working distance, camera/projector resolution, and calibration quality.

We accomplish such a sampling by rendering the object from multiple points of view with the virtual camera introduced above into a *geometry buffer* that stores object position and normal in object space for each pixel that the renderer generated. The virtual camera is thereby moving on a sphere centered at the object center with a radius that roughly equals to the later working distance (cf. Figure 2). Limiting the working distance is not a restriction as the fixed focus of the projector only allows for sharp projections at a certain distance range.

After each rendered view, we read out the buffer and accumulate the sampled points. A final voxel filter operation with a voxel size of ε eliminates duplicate points and assures that our mesh is equally sampled. Our sampling strategy has also another advantage: Occluded areas, such as the inside of an object that might be modeled in the virtual data, are automatically sorted out by the renderer and will not create any sample points. Without an occlusion handling those areas would complicate the registration process, as reconstructed feature points that always lie on non-occluded areas could be wrongly matched to occluded points.

We want to point out that our concept of a virtual camera is not limited to triangulated meshes as long as the renderer supports writing to a geometry buffer.

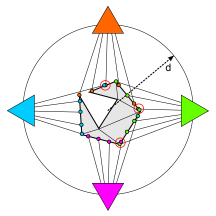


Figure 2: Sampling scheme in our point cloud creation process. Given a working distance d the virtual camera is placed on equally distributed locations on a sphere and the object coordinates are rendered to the *geometry buffer*. Note that some object points might be seen by two cameras at the same time (red areas). The voxel filter operation will eliminate these duplicates and assure equally distributed points.

4.4 Initialization and reinitialization

So far our incremental tracking algorithm assumed that the object pose in the previous frame was known. For the very first frame or when tracking is lost, a different object tracking strategy must be used. In detail, the correspondence matching of 4.1 is replaced by a combination of gray code patterns and a phase shifting technique to calculate a dense correspondence map. Although the physical object must remain static throughout the pattern acquisition (≈ 5 sec.), the density of the generated correspondence map and triangulated point cloud guarantees a convergence of the ICP to the correct object pose, provided that a good guess of the initial pose is given. Specifying an initial rotation manually and centering the centroid of the scanned point cloud with the virtual sampled point cloud automatically provided good results throughout our whole evaluation and does not burden the user with too much manual alignment. There is space for a lot of improvement at this stage. Basically any advanced coarse alignment method could be incorporated at this point. Using imperceptible patterns [3] encoded in the virtual model projection would not break the visual coherence. In future the dense point cloud would also allow to measure deviations between the real model and virtual model. The deviation could be incorporated in the tracking pipeline to further reject 3D points that do not fit the virtual model.

If tracking is lost during the continuous tracking phase, often it is sufficient to move the object to the last successfully tracked pose and therefore enable the ICP to converge again. This is in particular easy as the projector will show the virtual texture at that pose. The user can alternatively trigger the full initialization described above, if a manual adjustment is not possible for certain constellations.

5 Implementation Notes

Many parts of our algorithm are utilizing the parallel processing power of the GPU. We used the OpenCV GPU module that implements CUDA kernels for the corner detector and LK optical flow tracker. Image undistortion and predistortion, as well as the radiometric compensation, are realized by pixel shaders. Memory transfers between CPU and GPU can be costly, however, we could avoid them to a great extent in our approach since the rendered, predicted camera view is already in GPU memory. Only the captured image must be uploaded to GPU memory in each frame. Feature detection, feature matching and triangulation can be carried out completely on the GPU. Thereafter, the triangulated points must be transferred back to the CPU, as the current ICP implementation using the Point Cloud Library (PCL)[1] takes place on the CPU. Although using a kd-tree search structure, this operation is still the performance bottleneck of the whole application but could, similar to the approach in *KinectFusion* [6], be implemented for the GPU in future. Figure 3 summarizes our CPU-GPU implementation.

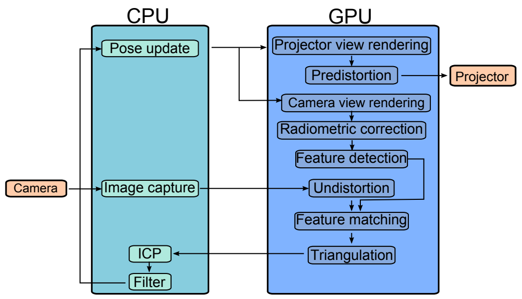


Figure 3: Dataflow diagram of our algorithm and distribution on CPU and GPU.

6 Experiments

To prove that our incremental tracking can robustly track objects with the projection of the virtual replication only, we carried out several experiments. In the following we briefly describe the evaluation setup and the scenarios that we have tested.

6.1 Hardware setup

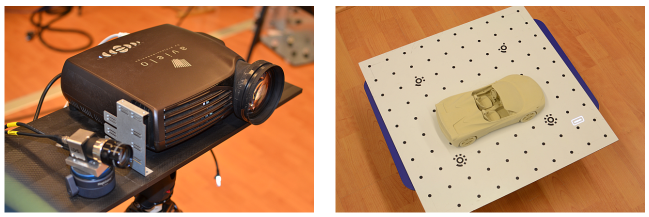


Figure 4: Left: Our mobile camera-projector system, mounted on a tripod. Right: Evaluation setup with marker board and temporarily fixed evaluation object.

For our mobile camera-projector system, we solidly mounted an IDS CMOSIS USB 3.0 grayscale camera with a resolution of 2048×2048 pixels and a Full HD ProjectionDesign Avielo video projector on a carbon board (cf. Figure 4 left). The baseline is approximately 40 cm and camera and projector were focused and calibrated for a working distance of around 1.5m. At this distance, the projector can illuminate a volume of around $50 \times 50 \times 50$ cm. The sampling distance in 4.3 was also set to this distance. We synchronized camera and projector by simply tapping the VGA V-Sync signal from a HDMI-to-VGA converter that was connected to an HDMI splitter. The other output of the splitter was directly connected to the projector. The tapped signal serves as trigger input to our camera. Note that due to the incremental nature of our tracking algorithm, hardware synchronization is not necessarily required but greatly increases the frame rate of our application since the additional "safety time" between projection update and image capture can be minimized. Our setup ran on an Intel Core i7 platform equipped with 32GB memory and an NVIDIA Quadro K5000 graphics card.

To evaluate the performance of our algorithm, we compared it with "ground truth" data created by an accurate marker tracking method that is running in parallel and is using the same camera. The marker tracker is based on bundle adjustment methods from photogrammetry and has submillimetric accuracy for a given aluminium board with printed circular markers. The test object was temporarily fixed to this board (cf. Figure 4 right). When moving the board around in space, the relative pose between our tracking result and the result from the marker tracker should remain constant.

6.2 Evaluation procedure

For each single test the board with the fixed model was initially placed at a location that was known to our incremental tracker so that the initialization step could be skipped. The system records both tracker poses and the number of features during each step of our algorithm while the marker board was moved manually inside the projection volume along its primary plane. To make results comparable, the offset between the car coordinate system and the marker board system was registered once before evaluation. The object velocity was limited so that tracking is not lost during the test.

We carried out the evaluation for one 3D printed model car with two different design variants. The first, quite artificial "Grid" design consists of a grid structure that was UV-mapped on the car model. This design inhibits strong corners and facilitates feature reconstruction. It allowed us to benchmark particularly the performance of our ICP algorithm. The second "Fancy" design contains a variety of random colored geometric primitives.

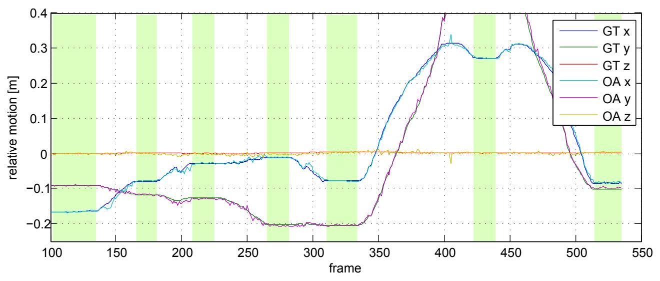
We also created a pseudo-animation for each design by simply shifting texture coordinates across the surface. To test robustness against outliers we placed modeling clay on the car so as to create partial, artificial deviations of the actual object from the virtual model. Table 2 lists all test cases in our evaluation.

6.3 Results

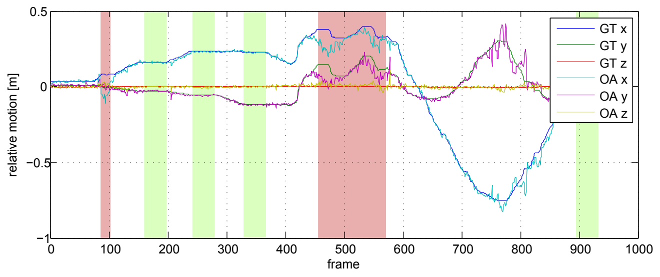
The recorded poses are shown in Figure 5. For the sake of clarity, we plotted the results over time only for the basic "Grid" and "Fancy" test case without animation and artificial outliers. For these scenarios, we also evaluated the difference between our tracking result and the marker tracking result and plotted mean and standard deviation in Table 1. It is clearly visible that the pose stability is higher for the "Grid" design during both translational and rotational movement. Looking at the recorded average feature numbers in Table 2 this seems reasonable, as far less features are finally taken for pose estimation in the "Fancy" case (cf. Figure 6). The more sparse the feature point cloud becomes, the less stable is the convergence behavior of the ICP algorithm. For the "Grid" case, an average error of around one to two pixel was measured based on the ground truth of the marker tracking method.

Further notable is the fact that our incremental tracker converges to the ground truth pose only after a certain number of frames, which is proportional to the object's velocity. This observation can be explained by the geometry of our car. With sudden, especially translational movements, large parts of the car become unlit, as the projection has not updated yet and consequently only features at the remaining lit part of the car are used for pose estimation in the next frame. Hence the estimated pose will not completely correct

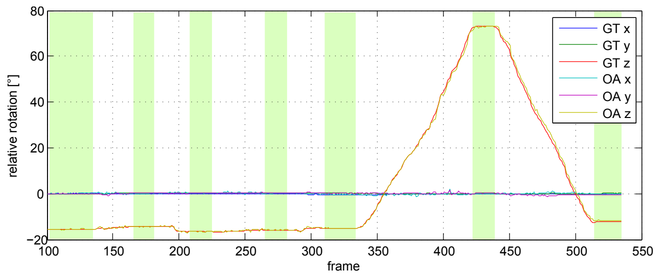
[1]`http://www.pointclouds.org/`



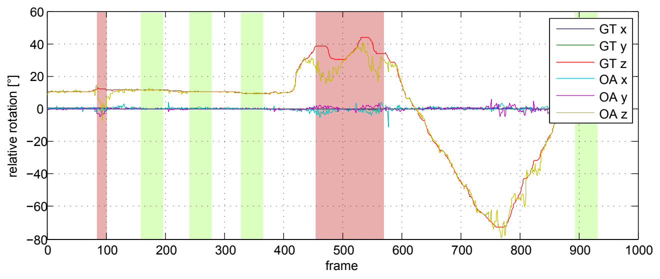
(a) “Grid”, translation



(b) “Fancy”, translation



(c) “Grid”, rotation (Euler angles)



(d) “Fancy”, rotation (Euler angles)

Figure 5: Motion curves for marker tracker (ground truth, “GT”) and our approach (“OA”) for the basic “Grid” design (a),(c) and the basic “Fancy” design (b),(d) test cases. Diagrams (a) and (b) plot the relative movement of the object’s center over time. Diagrams (c) and (d) show rotational movement along the object’s coordinate axes. The offset between car origin and marker board origin is already compensated. Note that the object was moved along the primary plane of the marker board only so that translation along Z and rotation around X and Y were almost constant. Green areas mark frames where the object was not moved. This data is evaluated separately in Table 1 (“Rest”). Clearly visible is the response time of our method after translational movement (for instance in diagram (a), frame 140-160 and 190-210). The ICP algorithm temporally diverged for the “Fancy” test case (red areas).

Table 1: Mean and standard deviation for measured absolute translation and rotation difference of our approach compared to ground truth data. Entries marked with “Rest” were computed by taking only values into account for which the object was not moved (green areas in Figure 5).

	Translation (mm)			Rotation (°)		
	x	y	z	x	y	z
“Grid” μ	3.2	4.0	2.1	0.23	0.18	0.42
“Grid” σ	3.5	4.7	2.3	0.23	0.23	0.58
“Grid” μ (Rest)	1.2	1.6	1.4	0.13	0.08	0.14
“Grid” σ (Rest)	0.9	1.4	1.3	0.13	0.14	0.15
“Fancy” μ	19.7	16.6	6.3	0.62	0.53	2.18
“Fancy” σ	31.6	27.1	7.4	0.79	0.75	3.93
“Fancy” μ (Rest)	4.2	4.0	3.5	0.33	0.15	0.31
“Fancy” σ (Rest)	4.5	3.5	3.3	0.30	0.12	0.44

the projection in one frame but increase the percentage of the lit surface part and enable more precise pose estimation in the consecutive frame (cf. Figure 5, diagram (a), frame 140-160). To account for this delay in the comparison of Table 1, we additionally evaluated the differences between both tracking results only when the object was not moved (green areas in Figure 5).

This iterative behavior can also fail under certain circumstances. In diagram b) and d) of Figure 5 the object movement at frame 85 causes an unfavorable constellation of feature points that let the ICP algorithm converge to a wrong pose. In the next frame, an even worse feature constellation is present and the pose diverges. After twenty frames, a good constellation is found and the pose approximates the ground truth pose again. Temporal divergence can also be observed between frame 450 and 565, which explains the high difference values in Table 1 in that case.

Turning on texture animation had different effects on the performance with respect to the used texture. For the “Grid” design, the number of features only marginally decreased compared to the case with no animation. Also the tracking performance was comparable. However, feature numbers dropped notably for the “Fancy” design. Only 20% of the detected features were finally used for pose estimation and led to noisy pose updates. The reason for the bad performance is not the motion of the texture by itself but the fact that through animation sometimes texture parts with less features are exposed to the camera. As long as there are enough features in every frame such as in the “Grid” case, there is no decrease in performance observable.

The outlier test based on the car model deformed by modeling clay (cf. Figure 7) also showed comparable tracking performance in the “Grid” case. The effectiveness of RANSAC in our ICP algorithm is clearly visible, rejecting about 17% of the reconstructed features (cf. Table 2, first row). For the “Fancy” test case, tracking performance was still acceptable although the RANSAC method was less effective.

Concerning tracking speed, we achieved an update rate of around 10 Hz with the current system setup. As mentioned above, most of the time is consumed by the ICP algorithm, which is currently implemented on the CPU and consuming approximately 50% of the overall processing time. By porting it to the GPU, we expect a performance boost that would make our approach realtime-capable.

7 Conclusion

In this work, we have presented a new tracking approach for the shader lamp concept that works with the projection of the virtual content only. Our system is mobile and does not require any modifications to the scene, the tracked object or the virtual projection. By exploring a new combination of feature reconstruction provided by [7] and point cloud registration, our system can track a moving

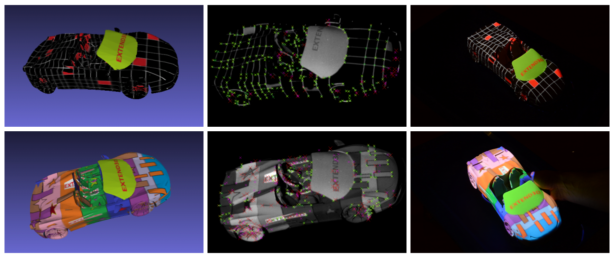


Figure 6: Our car model with the “Grid” (top) and “Fancy” (bottom) design. The first column depicts the virtual model with the texture applied. The second column shows the captured image along with detected and triangulated (green) and detected but finally rejected feature point pairs (magenta and red). The last column shows pictures of the lit object taken from an external camera. It is obvious that the texture has a huge impact on the performance of the feature matching.

Table 2: Average number of features during detection (D), triangulation (R) and point cloud registration (M) phase, absolute and relative to the number of detected features.

Test case	D	R	M	R %	M %
“Grid”	250	177	162	0.70	0.65
“Grid” animated	249	173	159	0.70	0.64
“Grid” outlier	248	150	125	0.60	0.50
“Fancy”	254	75	65	0.29	0.26
“Fancy” animated	214	49	42	0.23	0.20
“Fancy” outlier	250	65	61	0.26	0.24

object and “stick” the projection to it interactively.

We verified our approach quantitatively in a realistic 3D-printing scenario with different textures, animation and outlier simulation. We believe it is relevant also for applications in fields such as industrial design, rapid prototyping and AR showcases. The full potential of the method has by far not been exploited yet. Still, there are two major limitations to our approach: Firstly, the texture that is projected, must inhibit enough distinct features which might not be fulfilled for some shader lamp scenarios. Secondly, the physical model itself must provide enough surface details for a unique matching. Almost planar surfaces for instance may not be suited for our ICP registration approach.

In future, we will concentrate on the feature detection pipeline which is currently very basic and could for instance incorporate color in the feature description to improve performance. Furthermore, both frame rate and latency could be reduced by a faster, GPU-based implementation of the ICP algorithm. At the same time, this would also increase robustness against faster movements. Advanced filtering and extrapolation concepts would further increase the robustness. Lastly, the sparse ICP registration concept could be extended by more sophisticated 3D feature matching concepts that explore the spatial structure of the virtual model at a higher extent and allow for tracking of even more complex objects.

In conclusion we believe that our approach is very promising and could in the future offer a new kind of interactivity in the SAR context.

Acknowledgements

We would like to thank Beatriz Frieden and Andres Sanchez for their invaluable help during the assembly of the hardware components. This work was supported by the German Federal Ministry of Education and Research (ARVIDA project, grant no. 01IM13001N).

References

[1] E. Akaoka, T. Ginn, and R. Vertegaal. Displayobjects: prototyping functional physical interfaces on 3d styrofoam, paper or cardboard models. In *Proceedings of the fourth international conference on Tangible, embedded, and embodied interaction*, pages 49–56. ACM, 2010.

[2] D. Bandyopadhyay, R. Raskar, and H. Fuchs. Dynamic shader lamps: Painting on movable objects. In *Augmented Reality, 2001. Proceedings. IEEE and ACM International Symposium on*, pages 207–216. IEEE, 2001.

[3] D. Cotting, M. Naef, M. Gross, and H. Fuchs. Embedding imperceptible patterns into projected images for simultaneous acquisition and display. In *Mixed and Augmented Reality, 2004. ISMAR 2004. Third IEEE and ACM International Symposium on*, pages 100–109. IEEE, 2004.

[4] J. Ehnes, K. Hirota, and M. Hirose. Projected augmentation - augmented reality using rotatable video projectors. In *Mixed and Augmented Reality, 2004. ISMAR 2004. Third IEEE and ACM International Symposium on*, pages 26–35, Nov 2004.

[5] R. I. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, ISBN: 0521540518, second edition, 2004.

[6] S. Izadi, D. Kim, O. Hilliges, D. Molyneaux, R. Newcombe, P. Kohli, J. Shotton, S. Hodges, D. Freeman, A. Davison, and A. Fitzgibbon. 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, pages 559–568, New York, NY, USA, 2011. ACM.

[7] T. Johnson and H. Fuchs. Real-time projector tracking on complex geometry using ordinary imagery. In *Computer Vision and Pattern Recognition, 2007. CVPR’07. IEEE Conference on*, pages 1–8. IEEE, 2007.

[8] K. Khoshelham and S. O. Elberink. Accuracy and resolution of kinect depth data for indoor mapping applications. *Sensors*, 12(2):1437–1454, 2012.

[9] K.-L. Low. Linear least-squares optimization for point-to-plane icp surface registration. Technical report, Department of Computer Science, University of North Carolina at Chapel Hill, 2004.

[10] C. Menk, E. Jundt, and R. Koch. Visualisation techniques for using spatial augmented reality in the design process of a car. *Computer Graphics Forum*, 30(8):2354–2366, 2011.

[11] D. Molyneaux and H. Gellersen. Cooperatively augmenting smart objects with projector-camera systems. In *Proceedings of 3rd IEEE International Workshop on Camera-Projector Systems (ProCams 2006)*, pages 13–14, 2006.

[12] J.-N. Ouellet, F. Rochette, and P. Hebert. Geometric calibration of a structured light system using circular control points. In *3D Data Processing, Visualization and Transmission*, pages 183–190, 2008.

[13] R. Raskar, J. van Baar, P. Beardsley, T. Willwacher, S. Rao, and C. Forlines. ilamps: geometrically aware and self-configuring projectors. In *ACM SIGGRAPH 2006 Courses*, page 7. ACM, 2006.

[14] R. Raskar, G. Welch, K.-L. Low, and D. Bandyopadhyay. Shader lamps: Animating real objects with image-based illumination. In *Rendering Techniques 2001*, pages 89–102. Springer, 2001.

[15] R. Raskar, R. Ziegler, and T. Willwacher. Cartoon dioramas in motion. In *ACM SIGGRAPH 2006 Courses*, page 6. ACM, 2006.

[16] S. Rusinkiewicz and M. Levoy. Efficient variants of the icp algorithm. In *3-D Digital Imaging and Modeling, 2001. Proceedings. Third International Conference on*, pages 145–152, 2001.

[17] J. Underkoffler. A view from the luminous room. *Personal Technologies*, 1(2):49–59, 1997.

[18] R. Yang and G. Welch. Automatic and continuous projector display surface calibration using every-day imagery. In *Proceedings of 9th International Conf. in Central Europe in Computer Graphics, Visualization, and Computer Vision WSCG*, 2001.

[19] F. Zheng, R. Schubert, and G. Welch. A general approach for closed-loop registration in ar. In *Mixed and Augmented Reality (ISMAR), 2012 IEEE International Symposium on*, pages 335–336, Nov 2012.